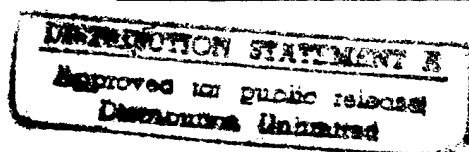19970502 241

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

DTIC QUALITY INSPECTED 4

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/96M-08

PROCEDURES FOR TESTING
DETERMINISTIC SCHEDULING MODELS:
A DAKOTA CASE STUDY

THESIS

Keith H. McCready, Captain, USAF

AFIT/GOR/ENS/96M-08

PROCEDURES FOR TESTING
DETERMINISTIC SCHEDULING MODELS:
A DAKOTA CASE STUDY

THESIS

Presented to the Faculty of the Graduate School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of Master of Science Operations Research

Keith H. McCready, B.S., M.S.

Captain, USAF

March 1996
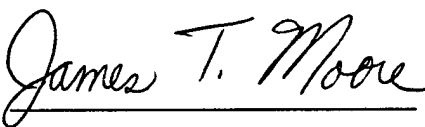
THESIS APPROVAL

NAME: Keith H. McCready, Capt, USAF   **CLASS:** GOR-96M

THESIS TITLE: Procedures for Testing Deterministic Scheduling Models: A DAKOTA Case Study

DEFENSE DATE: 18 March 1996

COMMITTEE	NAME/TITLE/DEPARTMENT	SIGNATURE
Advisor	Dr. Richard F. Deckro Professor Department of Operational Sciences	
Reader	James T. Moore, Lt Col, USAF Associate Professor Department of Operational Sciences	

Acknowledgments

Table Of Contents

List of Tables

AFIT/GOR/ENS/96M-08

Abstract

The DAKOTA scheduling system has been proposed for use in the United States

Air Forces Europe's (USAFE's) Operational Support Airlift (OSA) scheduling. This

thesis examines the OSA scheduling topic and reviews the relevant literature on vehicle

routing, concluding that exact methods are intractable for large problem sizes.

Consequently, heuristic methods must be considered. This thesis takes a detailed look at

the DAKOTA heuristic. It examines the concepts of Validation, Verification and

Accreditation (VV&A), particularly as they apply to heuristics and algorithms. It then

defines what measures of performance may prove useful in judging heuristics and

algorithms in general, and details the statistical tests which can be used to make those

comparisons. It discusses four of the predominant airlift scheduling models currently in

use, and finally develops a methodology which can be used to evaluate a deterministic

passenger airlift scheduling heuristic, using DAKOTA as a case study.

PROCEDURES FOR TESTING DETERMINISTIC SCHEDULING MODELS:

A DAKOTA CASE STUDY

I. Introduction

Background

Prior to 1989, the Military Airlift Command (MAC) was responsible for most airlift

missions in the United States Air Force. With the reorganization of MAC into the Air

Mobility Command (AMC), AMC retained the strategic airlift mission while the majority

of the theater airlift was divested to the theater commands. The United States Air Forces

Europe (USAFE) picked up the roles of theater airlift and Operational Support Airlift

(OSA) in the European theater. With these new responsibilities, HQ USAFE/DON

(Operations Analysis) found that the Global Decision Support System (GDSS) command

and control system was satisfactory for the scheduling of theater airlift missions (which

primarily consisted of cargo missions), but GDSS was less appropriate for OSA. While

investigating alternative means of scheduling the OSA missions, DON found that the Air

Force Office of Scientific Research (AFOSR) had supported a grant to North Dakota

State University (NDSU) to develop a logistics airlift (Logair) system. Based on this

information and other considerations, DON arranged for AFOSR to extend that grant to

develop a scheduling system for OSA airlift, known as DAKOTA.

USAFE requested an automated system to assist in scheduling OSA passenger support. The system was to facilitate the scheduling of critical mission travel throughout Europe, the former Soviet Union, and Africa, as well as intercontinental flights to the United States. USAFE has a heterogeneous fleet of aircraft, including C-20, C-21, and T-43 airplanes. This fleet has differing capacities, endurance, and speed, with each aircraft allocated a fixed number of flying hours. The objective of the automated system is to maximize overall efficiency of the fleet, supporting the travel of as many eligible personnel as possible with the least amount of flight time, within the constraints of fleet size, total flying hours available, and other limitations.

Currently, USAFE has twelve OSA aircraft available and schedules approximately six or seven missions each day. In order to schedule the OSA support, USAFE employs three schedulers on a full-time basis at its headquarters, with additional schedulers working at the wing and squadron levels.

The DAKOTA software accepts requests for travel and information about airports, aircraft, and passengers. Missions can be scheduled manually, or automatically by utilizing the automated decision support module. A number of tools are provided in DAKOTA to facilitate the scheduler's task, including maps, tables, feasibility checking, a "greaseboard," and summaries of available flying hours. A relatively large database is involved for both relatively static information (airport runway lengths, hours of operation, aircraft capacities and aircraft operating characteristics) and dynamic information (passenger and aircraft status). DAKOTA's optimization routine supports the work of the

scheduler by accepting the input sets of travel requests and missions, and producing optimal and/or near-optimal candidate schedules for consideration by the scheduler.

DON projects that successful implementation of DAKOTA may ultimately lead to its incorporation into GDSS for use in scheduling for the cargo mission as well. Additionally, the 89th Airlift Wing (ALW), responsible for presidential and DV (Distinguished Visitor) airlift support, is interested in the success of DAKOTA and its potential for their use. According to USAFE/DON, no other airlift scheduling models in use in the Air Force today have a built-in optimization routine. (Wilkinson, 1996) In a fiscal environment of tightening budgets, a scheduling system which schedules OSA more efficiently has great promise.

USAFE's OSA Mission

USAFE's 76th Airlift Squadron (AS) has nine C-21s, two C-20s, and one T-43 to support the OSA mission. (Maher, 1996) The nine C-21s, which are small executive jets with approximately four hour endurance, are used to transport qualified personnel to different locations when commercial transportation either is not available or is not cost effective. On average, USAFE schedules approximately six "lines", or missions, per day with the C-21 fleet, with three or four "sorties", or flights each, and keeps one additional aircraft on alert for short-notice or priority tasking (for instance, in support of the Bosnian peace efforts). Additionally, USAFE projects that for most of the remainder of the year, they will have one of their nine C-21s unavailable, as those aircraft are sequenced through a maintenance/modification cycle.

The C-20 is a larger executive jet, capable of carrying 12 passengers, which USAFE uses primarily to transport their most senior officers. The C-20 is reserved primarily for the theater's top generals for several reasons: 1. It has longer endurance, being able to fly up to eight hours unrefueled, making it an ideal candidate for flights from Ramstein Air Base, to Gander (Newfoundland), and on to Andrews AFB, Maryland. 2. It has a better communications suite, and its crew includes a dedicated Radio Operator. Among the communications equipment are the very long-range HF and SATCOM radios essential for the support of the senior officers. 3. USAFE has only two C-20s. Due to their larger size, USAFE may on rare occasions use C-20's to carry larger groups, particularly when it is not cost-effective for them to travel by commercial conveyance. Their C-20s fly an average of a single mission per day. (Lopez-Velasquez, 1996)

The final aircraft in the USAFE OSA inventory is the T-43, which is a converted B-737 that can carry 55 passengers. Due to its size and 5.5 hour endurance, it is primarily used for larger groups and longer ranges. The T-43 flies an average of only once per week.

Scheduling

USAFE schedules two primary types of OSA missions. One is "Out-and-Back" missions or day trips which depart their home base of Ramstein, fly to their destination, and then return to Ramstein on the same day. The other is a "Remain Overnight" (RON) flight where the airplane and crew spend the night at a destination away from home and then return home on a later date (typically the next day). USAFE generally prefers to schedule Out-and-Back missions, as those are easier for maintenance to support and less

4

taxing on the crews. A RON will often be scheduled when the distance traveled makes "deadheading," or flying an empty aircraft back to Ramstein only to return to pick up the same passenger(s) later ineffective. RONs may also be used when the return time the next day is indefinite. On average, about 65% of the missions are Out-and-Backs; 35% are RONs.

If USAFE gets a short-notice request for OSA, schedulers first look at their schedule to see if the request is supportable. If it is not, they then contact other US agencies which may be able to support the request, such as Army and Navy units, or the European Command (EUCOM). If none of these agencies can accommodate the request, the schedulers then consider getting approval to add another mission to the schedule. If approval is granted, they schedule the mission and the crew, and notify the 76th AS of the change. While it may seem an intimidating task, the schedulers report that the process of checking with the other organizations and then adding a new line takes approximately 30 minutes. (Lopez-Velasquez, 1996)

Whenever possible, the schedulers attempt to "aggregate" multiple airlift requests onto a single mission. Not only would eliminating the need for an additional aircraft to accomplish the same support allow more time for maintenance on the aircraft, but it would also reduce the number of crews required. Though USAFE does not currently have *excess* crews, it is expected that by the summer of 1996 there will be fewer crews available, highlighting the need to aggregate missions whenever possible. (Lopez-Velasquez, 1996)

Surprisingly, optimizing their schedule in order to reduce the number of flight hours needed to provide a given level of support may not currently be a high priority for USAFE. In fact, Maj Maher pointed out that in FY95 they ended up "turning in" unneeded flight time: approximately 300 C-20 hours, 200 T-43 hours, and 1700 C-21 hours. Part of the reason for the excess time remaining was the requirement to maintain an alert aircraft and crew in support of the Bosnian peace effort. (Maher, 1996)

The job of scheduling OSA for USAFE involves assigning crews to aircraft, allocating missions to requests, and coordinating with maintenance for the aircraft availability. Arranging the diplomatic clearances for required border crossings is generally not as serious a problem as it once was, since virtually all European nations now allow overflight. Switzerland has granted blanket overflight to DV aircraft to support the humanitarian mission in the Balkans, and even Austria allows passage with enough advance notice.

Scope

We have introduced USAFE's OSA mission, and briefly discussed the problems associated with scheduling passenger airlift. We will see that as problem size increases, exact solutions to the scheduling problem become intractable. Consequently, we find we need to rely on heuristic, or approximate, methods to solve the problem. The issue then becomes one of deciding how to determine if a heuristic provides the solutions that we seek.

Chapter II reviews the relevant literature on vehicle routing and takes a detailed look at the DAKOTA heuristic. It makes a concentrated examination of the concepts of

Validation, Verification and Accreditation (VV&A), particularly as they apply to heuristics and algorithms. It then defines what measures of performance may prove useful in judging heuristics and algorithms in general, and details the statistical tests which can be used to make those comparisons.

Chapter III provides a discussion of four of the predominant airlift scheduling models currently in use, and Chapter IV develops a methodology which one can use to evaluate a deterministic passenger airlift scheduling heuristic, using DAKOTA as a case study. Chapter V presents conclusions and recommendations for further research.

Finally, the appendices itemize several problems encountered in installing and running the DAKOTA program, and offer suggestions for improvements to the program.

II. Literature Review

USAFE's Operational Support Airlift scheduling problem is closely related to the class of problems known as the Vehicle Routing Problem (VRP). An entire area of study has been dedicated to trying to find better, more efficient ways to apply solution techniques to this class of problems. Though they are not the exact scheduling problem faced by an OSA scheduler, a basic understanding of the problem will provide important insights into the problem.

This chapter addresses the solution approaches many authors have designed for the Traveling Salesman Problem and other variants of the VRP. Its intent is to demonstrate the extreme difficulty of solving this type of problem to optimality. The consequence of the complexity of these problems is that many authors have developed *heuristic* methods of solving the problems, concentrating on trying to reach *good* solutions which use far less computational effort, instead of necessarily trying to find an optimal solution. The question then becomes not "What is the answer?", but instead "Is this a good enough answer?" and "What is meant by 'good'?"

This chapter considers how to compare the heuristics which have been developed to solve the vehicle routing problems. It starts by introducing some of the variants of the VRP and some of the solution techniques which have been proposed for them. It then reviews the heuristic incorporated within the DAKOTA scheduling program. Next, it examines how one might seek to validate, verify and accredit a heuristic, by considering

8

what the appropriate performance measures for a heuristic might be in the case of the USAFE OSA scheduling problem. Finally, it discusses what the applicable statistical tests for conducting the comparison might be.

Traveling Salesman Problem

The most basic, most widely studied case of the vehicle routing problem is the Traveling Salesman Problem (TSP). The TSP is defined by a set of nodes, the set of arcs connecting those nodes, and the costs associated with moving from node to node along the arcs. The TSP consists of a single vehicle, departing from its origin (or depot), visiting (or servicing) all of the nodes and/or arcs in the problem once and only once, and then returning to its origin. There is no restriction on the order any of the nodes are visited. The objective is to minimize the distance traveled or cost of traversing the arcs of the route.

If the specific problem requirements are tractable, an exact solution to the TSP can be determined by integer programming (IP). Unfortunately, the Traveling Salesman Problem is classified as NP-hard, indicating that an exact solution algorithm which is polynomially bounded is unlikely. (Lawler, Lenstra, Rinnooy Kan, and Shmoys, 1985:11) The repercussion of this classification is that as the network size increases, the storage requirements and solution times required grow exponentially. Lawler, et al., found that a threshold for the size of tractable problems using most solution techniques seems to be about 100 nodes, though a 318-node problem has been solved to optimality, and some think that 1000-node or larger problems might be solved exactly with modern

techniques. (Lawler, et al., 1985:15) In order to overcome the complexity of solving the TSP, many heuristic solution approaches have been developed.

Vehicle Routing Problem

Another routing problem, and one more closely related to the Operational Support Airlift scheduling problem, is the Vehicle Routing Problem (VRP). The VRP consists of several elements: (Swenson, 1986:13; Russell 1995:156)

- A set of geographically distributed customers who require a pickup or delivery exactly once.

- At least two vehicles (the one-vehicle VRP is the special case of the Traveling Salesman Problem) which originate and terminate at a central depot.

- The operation of the vehicle incurs some cost.

- In the Capacitated VRP, each vehicle has a finite capacity, and each customer places some known demand on the vehicle, which may vary from vehicle to vehicle.

- In the VRP with Time Windows, the customers must be serviced during some permitted pickup or delivery time intervals.

The VRP is an extension of the Traveling Salesman Problem which allows for more than one vehicle, which does not restrict the number of visits to each node, which can limit the capacity of each vehicle, and which places constraints on time. The optimal

solution of the VRP minimizes the cost of satisfying all the customers' requirements without exceeding any of the vehicle capacity constraints.

In the classical VRP, a constraint may be added to restrict vehicles to only one visit per node, but such a restriction is unlikely to apply to OSA's airlift scheduling problem. It will therefore not be included in our discussion of the VRP.

Solution Approaches

Swenson (1989) offered two general methods of approaching the solution of the VRP:

1. Exact Methods. Swenson reviewed an exact solution technique using branch and bound methods with Gomery cuts, which is applicable on problems with up to 60 stops. She also examined a method using Lagrangian relaxation in a branch and bound algorithm to solve a restricted VRP which can solve problems with up to 151 stops, minimizing the number of vehicles required.

2. Heuristics. For most problems larger than those, however, exact solution methods have not been found. One wishing to solve a more complicated problem will have to rely on using a heuristic method instead. Swenson suggests most of the heuristics for the VRP fall into five categories:

   a. **Tour building heuristics**. Stops are added one by one to the routing until all stops have been covered.

   b. **Tour improvement heuristics**. Starting with an initial set of feasible routes, the route is improved by removing and replacing arcs.

c. **Two-phase method**. The routes are built and extended in two phases iteratively, building the tour by adding arcs and then resolving the problem.

d. **Cluster-first, route-second method**. Stops are grouped into sets (clusters) to be served by a single vehicle, and then the routing within each cluster is determined by traveling salesman heuristics. The groups are determined by associating all stops within a certain radius of some "seed"; much research has been spent on deciding the optimal placement of the seeds and the group size.

e. **Interactive routing**. A human interacts in the partially-automated process, primarily to ensure that constraints which have not been modeled are satisfied.

Vehicle Routing Problem with Time Windows

A variant of the VRP is one in which the pickups and deliveries must occur during certain time windows. Swenson looked at the Vehicle Routing Problem with Time Windows (VRPTW), and concluded that the best solution method was an iterative approach. She feels that these solution methods fall into three classes: (Swenson, 1986:18)

1. Limit the problem. Some solution methods overlook the time constraints, allowing an exact solution to a limited problem.

2. Cluster first, ignoring the time constraints, then route within the clusters according to the time constraints.

3. Route building heuristics with the time constraints limiting the route segments which can be added to the route.

Her work concentrated on the second method: clustering first and routing second. Geographically proximal stops are grouped into clusters without considering the time constraints. The routing step heeds the time constraints, but is not bound by them. A feedback procedure then evaluates the routes according to both route length and the time constraints. If needed, the clusters are revised to account for the time constraints. The iterations are repeated until the stopping criterion (satisfactory routing or maximum number of iterations) is reached.

Swenson was concerned that most of the success of the method depended on the initial choice of the "seeds" used in determining the grouping of stops into the clusters. Additionally, she stressed concern for the added complexity caused by using the feedback loop to revise the clusters when the time constraints are not met. The VRP is complex enough, and while putting it inside a feedback loop may sound drastic, she could see no way to avoid considering the time constraints inside the initial clustering.

One of the more promising developments for solving the VRPTW was detailed by Russell. He points out that in most real-world problems, there exists more than one objective. He rank orders the most common as: (Russell, 1995:156)

1. Minimize the number of vehicles (routes)

2. Minimize the schedule time

3. Minimize the travel distance

Most of the early work on VRPTW included both exact and heuristic methods. Exact methods attempted included branch-and-bounding, column generation, and k-tree relaxation approaches, which Russell states have been able to solve to optimality

problems with up to 100 customers who wish to travel among 25-50 nodes. However, Russell argues that efficient heuristic methods must be used for most of the real-world problems. (Russell, 1995:157) For example, Solomon (1987) proposed a VRPTW heuristic which had a primary objective of reducing fleet size required.

Russell's contribution was an examination of a parallel insertion heuristic and the choice of seed points, the order of insertion of the points, and the post-processing of any unrouted customers. He then examined a local search interchange heuristic which can improve upon the initial feasible solution to the VRPTW while limiting its search to within a neighborhood. He found that if the neighborhood is restricted to be too small, then the search fails to result in any improvement to the solution; if the neighborhood is too large, the problem becomes intractable—the classic neighborhood search dilemma. Russell's proposal was a hybrid approach: Instead of applying the interchange heuristic to find an improvement to an existing feasible solution, he embedded it within the construction phase of the initial feasible solution. He applied his hybrid heuristic to a large variety of test problems from the literature (Russell, 1995:161-165), and compared his results with those from parallel seed point, interchange, and k-opt two-phase and hybrid methods. While his heuristic was not the fastest he examined, he reports it generally yielded the best solutions to the VRPTW, and dominated the other approaches in terms of the number of vehicles (routes) in the solution with a 6.8 percent average reduction.

A heuristic for the VRP with Soft Time Windows was developed by Koskosidis, Powell and Solomon (1992). They express it as a mixed integer program, and use a

heuristic to extend the cluster-first, route-second algorithm for its solution. Koskosidis, et al., formulated a method of treating the time windows as "soft constraints" which are penalized a cost in the objective function when violated, hence the name soft time windows. Customers with tight delivery time requirements receive a high weight in the penalty function. The heuristic solves the problem in two phases: clustering the assignments, and then routing and scheduling among the clusters. They claim one of the advantages of this approach is that, since soft window constraints are more flexible, the heuristic is less likely to be caught in local optima. Koskosidis, et al., report their heuristic performed favorably compared to contemporary local insertion and improvement heuristics, and that it is capable of finding solutions in cases when a hard time window formulation would have failed.

General Pickup and Delivery Problem (GPDP)

In this general problem, vehicles have to transport loads from origins to destinations without transshipment at intermediate locations. Savelsbergh and Sol (1995) describe three special cases of the GPDP:

1. The Pickup and Delivery Problem (PDP), where each request has a single origin and a single destination, and all vehicles originate from and return to a central depot.

2. The Dial-A-Ride Problem (DARP), where all loads are generally of unit size.

3. The VRP, where either all the origins or all the destinations are at the depot.

They attempt to generalize these three cases to deal with many of the practical pickup and delivery situations by considering that the transportation requests often are received while the vehicles are "on the road," so the concept of depots is moot. They examine several different goals represented in the objective functions:

1. Minimize duration of the time a vehicle needs to execute the route.

2. Minimize completion time of the latest service activity.

3. Minimize time spent traveling between locations.

4. Minimize route length.

5. Minimize client inconvenience, or how nearly the requested pickup and delivery times are met.

6. Minimize the number of vehicles, which is nearly always the objective in the DARP.

7. Maximize profit, where the dispatcher does not have to fulfill all transportation requests, but can reject those deemed unprofitable.

Savelsbergh and Sol discuss solution approaches, dividing them into static and dynamic classes, and single- and multiple-vehicle problems. They point out that in a VRP most transportation requests which are geographically proximal are served by the same vehicle, since the vehicle generally originates at a single location (the origin). In the PDP, however, even geographically close destinations may be served by different vehicles, since those vehicles may come from different origins. They then discuss exact and approximate solution methods for each type of problem.

Dial-a-Ride Problem

The Dial-a-Ride Problem concerns scheduling a vehicle to transport a customer from a specified pickup location to a specified delivery location. This is similar to the OSA scheduling problem, with the added constraint of only one passenger per travel request. Several variations on the DARP include a many-to-many problem, involving multiple customers and pickups from and deliveries to a location unique for each of those customers; advance request systems, which permit the requests to be processed and scheduled in advance in batches; and immediate request systems, which require the requests to be scheduled by an insertion method.

Solutions to the DARP have been an area of significant research. Psaraftis considered an exact dynamic programming solution for the single vehicle problem. His algorithm requires an exponential increase in solution time and storage for an increase in problem size, and has only been solved for relatively small problems of up to 10 nodes. He noted the necessity of using heuristics for the solution of real problems. (Psaraftis, 1983:356)

Further refinements to the problem include Psaraftis' k-interchange algorithm, where k tour links of a tour are exchanged for k other links, which Jaw (1986) found useful as a subroutine in his multi-vehicle advance-request algorithm with time windows. Jaw followed up on Psaraftis' work by developing a heuristic solution to the multi-vehicle with time windows problem, with an insertion technique which accounted for pickup and delivery time windows. Jaw's method was to order the requests prior to scheduling according to their requested pickup time.

One of the disadvantages to these methods is that they take a myopic outlook; by assigning requests sequentially, the "greedy" method may overlook a more preferable global solution. Walker (1994) described an alternative solution technique to attempt to compensate for that potential shortcoming. That algorithm included an objective function which penalized minor violations of the constraints without ruling them out completely, in an attempt to permit potential solutions which may be "slightly infeasible" but may otherwise be acceptable.

Solanki and Southworth (1991) suggested a heuristic specific to an airlift problem, which accounted for the capacities of the airfields, and the rate at which they could service the arriving and departing flights. Their paper described their work on the Airlift Deployment Analysis System (ADANS) project. Their work was funded by the then-Military Airlift Command (MAC) to develop a means of updating the routes and schedules of aircraft transporting passengers and cargo when requirements change.

They use an insertion heuristic, starting with an existing schedule and adding new missions one at a time, solving more difficult problems iteratively. The object of their algorithm is to minimize a weighted sum of the tardy deliveries (i.e., those delivered after their Latest Arrival Dates). The weights given to particular deliveries enable the scheduler to prioritize missions. High priority cargo or passengers are granted a higher weight, forcing the algorithm to consider satisfying those requests over others. Similarly, placing a higher weight on outsized cargo forces the model to assign those missions to the large C-5 or C-17 aircraft which can handle the outsized loads.

According to the authors, the requirement stated by MAC was for the heuristic to solve a large problem consisting of some 500 to 700 aircraft, up to 200 airfields, and one week duration, in one hour or less of CPU time on a dedicated mainframe computer. They stress that they had only tested their method on problems of 42 aircraft and 10 airfields, but that they had considerable success since each used less than two minutes of processing on a SUN-4.

Solanki and Southworth assert that for larger problems, "The expected growth in computational effort with increasing problem size should be linear" for several reasons: (Solanki and Southworth, 1991:128-129)

1. Each insertion adds only (up to) two new airfields visited for an aircraft (the new origin and destination).

2. The computational effort to perform one insertion will be proportional to the small number of aircraft picked to be candidates in Step 1.

3. The number of insertions (and missions) will be roughly proportional to number of aircraft.

Therefore, the computational effort will be the product of the effort per insertion and the number of insertions—a linear function of the number of missions added.

An alternate approach to a more specific transportation problem was proposed by Psaraftis (1983). His algorithm allows for only one vehicle making deliveries from N distinct origins to N distinct other destinations. Psaraftis admits that such problems rarely exist in nature, but rationalizes that an efficient algorithm for a single vehicle problem can ably serve as a *subroutine* for a multivehicle problem, as in Solanki and Southworth.

In fact, that is the essence of the second step of the Solanki/Southworth algorithm, where the candidate aircraft (i.e., single vehicle) is chosen to serve the extra "inserted" mission, as well as its previous ones.

The difference is in the approach the two methods use. Psaraftis' algorithm "reshuffles" the origins and destinations, and reconsiders the entire problem from scratch. Thus, its required computational effort will be $O(N^2 3^N)$, and its storage requirement is $O(N3^N)$ (Psaraftis 1983:356). This algorithm is considerably more complex for larger problems than was the algorithm suggested by Solanki and Southworth. In fact, Psaraftis himself posited that "computational burden associated with an exact approach cannot be overly de-emphasized, and this has prompted many research groups to use heuristic routing algorithms instead of exact." (Psaraftis 1983:356)

One concern is that inserting missions one at a time will yield a "best place to fit the new mission in our existing schedule" schedule, but not necessarily a "best possible" (i.e., optimal) schedule. It can be concluded, as Psaraftis notes, that for even reasonably large real-world problems, the heuristic proposed by Solanki and Southworth will at least be tractable.

The approach described by Solanki and Southworth, that of minimizing a weighted sum of the tardy deliveries, is defined by Walker (1994:125-128) as "Minimum Fly Time Value Ordering." He provides an alternative, "Minimum (Soft) Time Violation Value Ordering," in which the goal is to minimize the *number* of soft time violations (i.e., the number of tardy jobs). Unfortunately, Walker explains that this approach requires a

20

linear programming problem be solved for each possible insertion; this will only be practical for problems of limited scope.

We have examined several variants of the Vehicle Routing Problem. In the VRP with Time Windows, we found that the time constraints require heuristics which either take advantage of the multiple objectives usually pursued (minimize the number of vehicles, the schedule time, or travel distance), or that solve the problem in two phases (cluster-first, route-second). Some treated time windows as "soft constraints", but penalized violations. We investigated General Pickup and Delivery Problems, which considered multiple different objective functions. Finally, we looked at Dial-a-Ride Problems and several heuristic solution methods proposed. In all cases, we found that exact methods were unable to cope with the computational effort required when problems grew larger, and most of the authors concentrated on heuristic solution methods. We are not guaranteed that these heuristics will produce optimal solutions to the problems posed, but they enable us to attempt larger, more complex problems and still hope for *reasonable* solutions. Certainly an *optimal* solution is our goal, and we are willing to accept a solution which is less than optimal only when necessary. Goldberg, whose recent work has centered on genetic algorithms (a heuristic approach), asserts that the operational goal of optimizing approaches, particularly for larger problems, is *improvement*, not necessarily *optimality*:

> The most important goal of optimization is improvement. Can we get to some good, 'satisficing' level of performance quickly? Attainment of the optimum is much less important for complex systems. It would be nice to be perfect: meanwhile, we can only strive to improve. (Goldberg, 1988:7)

We have looked at employing heuristics to solve several variations of the vehicle routing problems. Now we turn to look at one heuristic which was developed specifically for the USAFE OSA scheduling problem.

DAKOTA's Heuristic: Airlift Resource Allocation and Scheduling Problem (ARASP)

Walker looked at the application of heuristics to scheduling OSA airlift, and developed and documented the primary heuristic implemented in the DAKOTA scheduling system. In his development he introduces the Airlift Resource Allocation and Scheduling Problem (ARASP) classification, and describes the characteristics of the problem:

- Travel Request Characteristics

    - Origin

    - Earliest/Latest Pickup Times (Hard or Soft Times)

    - Destination

    - Earliest/Latest Delivery Times (Hard or Soft Times)

    - Priority

    - Contingent Size

- Resource Characteristics

    - Fleet Size/Type

    - Capacity

    - Speed

    - Endurance

- Operational Constraints

    - Length of time an aircraft may remain in service

    - Required maintenance schedule

    - Flight crew duty restrictions

- Performance Measures

    - Support as many requests as possible

    - Honor hard times

    - Penalize soft time violations

    - Minimize subsequent scheduling changes

The main core of the ARASP scheduling system is the SchedGen algorithm developed by Walker (1994). The SchedGen algorithm incrementally schedules travel requests onto aircraft missions defined by the scheduler. It represents feasible assignments of request legs to aircraft missions as columns for a set packing (SP) problem. It uses three primary sub-algorithms in accomplishing its objective; the first (*ColGen*) controls the order of SP column creation, the second (*ActFits*) determines *where* schedule insertions can be made to maintain feasibility, and the third (*Einsert*) decides *how* insertions should be made. Walker describes each of these functions in detail. (Walker, 1994).

For the discussions which follow, the definitions below will be used:

Mission: A specific aircraft and crew (and associated support) designated for a time period as available to provide support for airlift requests.

Request: An application for a (group of) passenger(s) to receive airlift support. A request can have multiple "legs", indicating the contingent of passengers wishes to travel to more than one destination.

Hard Time: A strict bound on a time window for a request (e.g., a traveler *needs* to reach a destination by a certain time).

Soft Time: A bound on a time window for a request which *can* be violated if necessary (e.g., a traveler does not *wish* to leave before a certain time, but would be willing to leave earlier if necessary).

Associated with each mission is a schedule, composed of an ordered list of events:

$$R = \{r_1, \ldots, r_n\} \tag{1}$$

where n is the number of request legs scheduled. When determining which requests to assign to missions, the requests can be *ordered* according to several decision rules (e.g., most constrained first, earliest pickup time first, or random), or ordered manually by the scheduler. Walker points out that "the performance of any insertion algorithm is inherently affected by the order in which the insertions are attempted and by the value associated with a particular insertion." (Walker, 1994:43) Insertion attempts for the requests follow the order presented in the sequence.

The table below gives a sample of a request with two legs, in which three passengers request round-trip travel between Ramstein Air Base, Germany (EDAR) and Laarbruch (EDUL):

Table 1

Sample Requests

Origin	Pickup Time			Destination	Delivery Time			# Pax
	Date	Early	Late		Date	Early	Late	
EDAR	2/1/95	(0645	0745]	EDUL	2/1/95	(0800	0830]	3
EDUL	2/1/95	(1100	1200]	EDAR	2/1/95	(1215	1245]	3

Open parentheses, as in "(0645" indicate soft time bounds, and square brackets, as in

"0745]" indicate hard time bounds. So "(0645   0745]" indicates the travelers *wish* to

depart no earlier than 0645, but *must* arrive *by* 0745.

An example of a mission which might be able to accommodate the requests from

Table 1 is:

Table 2

Sample Mission

Home	Early Time		Late Time		Aircraft			
Base	Date	Time	Date	Time	Type	Airspeed	Endurance	Capacity
EDAR	2/1/95	0600	2/1/95	2000	C-21	440	4.0 hrs.	7

In the SchedGen algorithm, the flight time is computed using the great circle

distance and the aircraft's airspeed. Legs which violate the endurance constraint require

insertion of an additional fuel stop. At each intermediate stop, a ground service time of

75 minutes is assumed for landing, unboarding, refueling, boarding and taking off—the

same layover time USAFE uses for their planning. Times for each event must consider

both its earliest permissible time and the completion time of all preceding events which

must be completed prior to the event under consideration. Event times and time window

constraints are propagated forward and backward using Critical Path Method techniques. Thus, "the earliest time for an event is the latest of the earliest time the event can take place and the earliest time which will satisfy the constraints of *all* request legs associated with the event," and "the latest time for an event is the earliest of the latest possible times for the event and the latest time which will satisfy the constraints of *all* associated requests legs." (Walker, 1994:41)

Description

The algorithm Walker described is as follows: (Walker, 1994:44)

Scheduling Process for ARASP

- Select candidate missions to schedule onto

    - Check out missions from scheduling database

    - Form set M

- Select request legs for scheduling

    - Check out legs from scheduling database

    - Form set R

- Order request legs in set R according to selected variable ordering strategy

- For each mission M_i, i = 1,...,m do

    - For each request leg r_j, j = 1,...,n do

        - Generate SP columns for representing feasible insertions of request legs from the subset $\{r_j,...,r_n\}$

- Use SP solver to heuristically determine columns representing the "best" assignment of request legs to missions

- Use insertion algorithm tools to recreate schedules for selected assignments

- Postprocess event list to obtain specific arrival and departure times

- Review / Alter schedules

- Check in missions and request legs and request database update

The objective may be minimizing cost or minimizing soft time window violations while maximizing the number of request legs supported.

After the heuristic proposes a solution, the human scheduler can accept or reject the solution, or can then schedule additional request legs onto the proposed solution or make other alterations. Once the schedule is finalized, the solution is saved to the database for implementation.

While the insertions may be determined by a greedy algorithm (such as *ActFits*), one that makes assignments without regard for future insertions, a better algorithm would be one which attempts to create a schedule which is most flexible for subsequent insertions. (Walker, 1994:48-49) One measure of flexibility is the size of the time windows for required events. Compressing the mission by delaying its origination until the latest permissible time and terminating as early as possible may produce a "tight" schedule, but one which is "brittle", vulnerable to errors or even small changes in the schedule.

Another factor which must be considered by the heuristic is constraint

propagation. Constraints may include limits on the number of flight hours on each

particular aircraft or aircraft taken out of service for routine maintenance or training

missions. (Walker, 1994:52) Expanding or contracting the time windows for events

cause changes which must be propagated; requests serviced by a mission leg influence

other legs, the mission, and the affected aircraft; onload or offload of passengers alters the

remaining capacity of the aircraft. In light of the latter concern, SchedGen provides for

"turning off" capacity checking, allowing the scheduler to observe performance, and to

then negotiate with the requester to support fewer passengers, when appropriate.

Computational Effort

The SchedGen heuristic must generate columns for feasible schedules for each

mission under consideration. A concern would be for the additional computational effort

required with growth in the problem size, but Walker asserts that "we can expect to

experience linear growth in computer CPU time to perform the generation." (Walker,

1994:69) This is comparable to the conclusion drawn by Solanki and Southworth in their

work on the ADANS project, where they determined that "the expected growth in

computational effort with increasing problem size should be linear." (Solanki and

Southworth, 1991:128-129)

Walker then discusses the computational effort involved in the growth of the

event list when insertions are considered. He justifies a worst case order of $O(n^3)$ for the

scenario with n request legs. (Walker, 1994:78) There are several rules of thumb

incorporated into the algorithm to reduce the size of the event list further. For instance,

no travelers are expected to travel through the same stop twice enroute to their destination. Also, if an event's latest time is earlier than a pickup's earliest time, or if an event's earliest time is later than a delivery's latest time, the search through the event list can be curtailed. Nevertheless, Walker concludes that, out of concern for the growth of computation times, it is in the interest of the human scheduler to limit the lengths of the missions and to provide reasonable constraints on the requests to allow the algorithm to complete the task in a reasonable time. (Walker, 1994:79)

Variable Ordering

Variable ordering concerns focusing efforts on the most critical decision points in solving the problem. Generally, insertion techniques suffer from the "myopia" inherent in committing to an insertion without fully considering the consequences. SchedGen attempts to compensate for the shortsightedness by establishing an ordering strategy for the insertion attempts. The ordering strategies considered are:

- Most Constrained First (MCF): Consider first those request legs which have the least likelihood of being scheduled

- Time Bound Variable Ordering: Sorting the request legs according to early/late pickup times (EPF/LPF), or early/late delivery times (EDF/LDF)

Walker's testing indicated that Earliest Pickup First (EPF) ordering consistently succeeded in improving schedule quality and reducing the necessary search. (Walker, 1994:81)

Value Ordering

Finally, once candidate schedules are produced, they must be ranked in order of how "good" they are. Walker described three different strategies: Minimum Fly Time (maximize the number of request legs supported while minimizing the marginal cost of the added support), Minimum Soft Time Violation (depending on the priority of the request leg), and Least Constrained Schedule (a robust schedule which can absorb minor changes while remaining feasible).

Interim Summary

In summary, we have seen that the transportation problem posed by the OSA mission is highly complex; solving routing problems becomes computationally difficult as the problem size increases. Several authors pointed out that "the mathematical programming problems arising from scheduling applications are often difficult to solve, and workable schedules must often be obtained from heuristics." (Sklar, et al., 1990:76; Bodin and Golden, 1981:97-108) For the traveling salesman problem, we saw that a threshold for the size of tractable problems using most solution techniques seems to be about 100 nodes. (Lawler, et al., 1985:15) Consequently, many authors have proposed heuristic solution techniques in an attempt to be able to find "good" solutions to the problem. Reeves defined a *heuristic* as "a technique which seeks good (i.e., nearly optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is." (Reeves, 1993:6) This leaves those who wish to use the heuristics with several issues to ponder:

1. When should use of a heuristic be considered, and when should finding the optimal solution be attempted.

2. If it is decided to use a heuristic, how does one determine *which* heuristic to use.

3. Once a solution is obtained, how to measure if it is a "good" solution.

4. What is meant by "good" solution.

This thesis next examines the concepts of validation, verification, and accreditation. The terms are introduced in the context most commonly found in the literature: modeling and simulation. We then consider how to apply these concepts to the task of evaluating heuristics and algorithms, proposing how one would validate, verify or accredit a heuristic or an algorithm. The characteristics which may be of interest, and means of statistically comparing the measures of interest are reviewed.

Validation, Verification and Accreditation (VV&A)

This section introduces the concepts of validation, verification, and accreditation as they are discussed in the literature. We consider particularly the notion of VV&A as it is applied to heuristics and algorithms.

Gass (1992) suggests that the purpose of validation is to convince the modeler "who should not need much convincing" and others "who should require a great deal of convincing" that the model would be of use to a decision maker. (Gass, 1992:250) His concept of the purpose of validation is to provide the decision maker some degree of "confidence" in the model. Since the confidence is from the point of view of the decision maker, not the modeler, the confidence is an attribute not of the model, but of the model

user. "We take an extreme position by saying that a decision model without a designated user (which implies a specific use) has no basis upon which a confidence statement can be made, that is, the *a priori* confidence level is zero." (Gass, 1992:250)

This leads into the concept of accreditation. Gass gives the definition of accreditation as "an official determination that a model is acceptable for a specific purpose." (Gass, 1992:251) The key to the definition is the "specific purpose" clause. Just as a model must be designed only for specific purposes, it must be graded or judged according to how well it achieves a specific purpose. If it fails to meet the standards fully, then Gass suggests that "the item in question can receive *limited* and *restricted* accreditation." (Gass, 1992:251) This again suggests the notion of accrediting the model with respect to its explicit specifications. Gass submits that the review needs to be done by an independent third party, with a specific user in mind, and should produce a report that gives guidance on whether or not the model can be confidently used for the intended purpose. By stating the criteria for the accreditation, the user implies some sort of weighting on those criteria in determining how well the model meets the overall objective.

Gass gives the following definitions: (Gass, 1992:253)

1. *Verification* is the process of determining that a model implementation accurately represents the developer's conceptual description and specifications.

2. *Validation* is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

He suggests that the independent review should judge the model according to several criteria: (Gass, 1992:254)

- Specifications: Were the specifications adhered to?

- Verification: Is the model reliable and an acceptable representation of what was supposed to be done?

    - Mathematical Logic

    - Computer Code

- Validation: Is the model a suitable representation of the decision situation?

    - Theoretical Validity

    - Input Data Validity

    - Operational Validity

    - Face Validity

- Pedigree: How capable is the model of solving the current problem based on its previous incarnations, past uses and the reputations of its developers?

- Past Uses

- Developer

- Configuration Management: Was the model development process managed properly?

- Usability: Can this model be used to solve the problem with the given resources?

- Documentation: Is the documentation sufficient to use the model?

Miser stresses that "a theory or model is an intellectual construct designed to approximate a selected aspect of reality." In that light, he offers the following definitions: (Miser, 1993:212)

1. *Validation* is the process by which scientists assure themselves and others that a theory or model is a description of the selected phenomena that is adequate for the uses to which it will be put.

2. *Verification* is the process by which scientists assure themselves and others that the actual theory or model that has been constructed is indeed the one they intended to build.

Miser contemplated the criteria to use in validation, and concluded two inferences: (Miser, 1993:213)

- There are no universal criteria for validation. Rather, the basis for judging the confidence that one should have in a model rests on the situation being modeled and the work that has been done, both in formulating the model and in comparing its consequences with reality.

- Any validity judgment is relative in at least two ways: with respect to the phenomena being modeled and the uses to which the model will be put. Thus a model may be used with adequate confidence in one situation but not in another.

Dery, Landry and Banville took an epistemological approach to the issue of model validation, phrasing it in the terms of the scientific method. Ultimately, they concluded that there can be no consensus found regarding the rules or process of validation: "There is no agreement either on what is a valid model or on what is *the* way to validate models." (Dery, Landry and Banville, 1993:168-169)

Landry and Oral concurred with this view when they wrote "...there is not one universal scientific method and therefore there cannot be a universal set of criteria for model validation." (Landry and Oral, 1993:162-164) They classified four types of validity:

- Formulational Validity: the degree of relevance of the assumptions and theories underlying the formal model of the managerial situation for the intended beneficiaries

- Experimental Validity: the quality of the solutions, the types of the solutions, the nature of solution techniques and the efficiency of the solution procedures

- Operational Validity: focus on the usefulness, timeliness, synergism, and cost of implementing the solutions provided by the model

- Data Validity: sufficiency, accuracy, appropriateness, availability, maintainability, reliability, and cost of data

They equated model *accreditation* with *credibility*, explaining that "In this context, an accredited model becomes a legitimate model for use; in other words, it is licensed to be used for the intended purpose." (Landry and Oral, 1993:166)

Oreskes, Shrader-Frechette and Belitz took a different approach in their discussion of verification, validation and confirmation of numerical models. Their definitions are reversed from those of most other authors. They defined verification as the demonstration of the truth of a model, implying its reliability as a basis for decision-making. Validation, to them, means not that the model is an accurate representation of physical reality, but rather a model that does not contain known or detectable flaws and is internally consistent. (Oreskes, Shrader-Frechette and Belitz, 1993:641-642)

Next, they considered the concept of "confirming" the model. Their assertion is that the best one can do in testing a model is to confirm that it produces the desired output. A model's failure to reproduce the expected outcome implies a fault in the model, but the reverse cannot be true; more confirming observations do not demonstrate the veracity of a model, they only support the *probability* that the model has no fault. (Oreskes, et al., 1993:643)

They judge that most of what passes for verification and validation is at best confirmation. Their concern about confirmation is that it is always a matter of degrees of probability, and that it encourages the modeler to claim positive results. Thus, they conclude that "models are most useful when they are used to challenge existing formulations, rather than to validate or verify them." (Oreskes, et al., 1993:643-644)

Rykiel (1994) criticized the Oreskes article, stressing that "validation" is equivalent to "acceptance testing", and with possible rare exceptions, validation and verification cannot be used to prove that the content of a model is "true." He makes four points:

1. Modelers should define validation and verification for the (technical) context in which they intended to use those terms.

2. Failure to do so will create a false sense of truth, instead of creating consensus.

3. Modelers should specify the context of the model.

4. They should "use model acceptability and performance indices" in describing testing results, instead of "declarations of validation."

Bacsi and Zemankovics (1995) suggest several statistical methods that can be used for assessing goodness of fit between model and field observations. They note that, especially for small sample sizes, results of the various tests can often be contradictory, and deciding whether a model performs well or badly is often subjective, with the statistical or mathematical analysis useful to support the subjective assessment. (Bacsi and Zemankovics, 1995:263)

Law and Kelton developed a three-step approach to developing valid and credible simulation models, which they feel "will not guarantee an absolutely valid model, but it will make the model more representative of the real system and also more credible." (Law and Kelton, 1991:307-314)

1. Develop a Model with High Face Validity. Develop a model which seems reasonable to those who are knowledgeable about the system.

   a) Conduct conversations with systems experts.

   b) If a similar model already exists, collect data and make observations of that system.

   c) Use relevant results from similar simulation models.

   d) Use results from existing theory which may govern the process.

   e) Use experience and intuition, especially in complex systems where some assumptions may need to be made.

2. Test the Assumptions of the Model Empirically. Test the probability distributions used in the model, and conduct sensitivity analysis on the output.

3. Determine How Representative the Simulation Output Data Are. Law and Kelton write that this is the most definitive test of a simulation model's validity. The data should be compared either to real-world output data, or to the data of other existing simulations.

Finally, Elmer examined several methods of validation, verification, and accreditation as they relate to Department of Defense regulations and instructions on the subject. He synthesized the work of Law and Kelton and several other authors (Balci, Sargent, and Davis) into a procedure, which he called the Proposed Integrated Methodology (PIM) model, described below: (Elmer, 1995:27)

1. Apply the definitions and concepts to communicate the important issues of VV&A to the customers

2. Determine tradeoff of cost vs. value of the confidence gained

3. Document all work in validation effort

4. Examine validity of data

5. Develop the model with high face validity throughout the entire building process, with system experts, experience and intuition, and Peer Reviews

6. Experimental design validation

7. Test or verify the assumptions made in the conceptual modeling

8. Test the model's output with empirical techniques, especially if historical data exist

9. Explain the process to the customer

His PIM combined the better points of works he investigated, combining them into a nine-step "checklist" for the user to follow.

In summary, most authors collectively agree that the purpose of VV&A is to verify that a model is appropriate and acceptable *for a specific purpose*. The implication is that VV&A is not an absolute guarantee of the model's suitability, but an expression of the decision maker's confidence that the model is apropos for the intended purpose: Does it meet specifications, is it an acceptable representation of what was intended, and is it a suitable portrayal of the decision situation. Next, we'll consider the concept of VV&A and extend it to the evaluation of heuristics and algorithms.

Evaluating Heuristics and Algorithms

Some authors have applied the concepts of validation and verification to testing heuristics and algorithms. One example is a report by Barr, Golden, Kelly, Resende, and Stewart (1995), which discusses the design of computational experiments and reporting guidelines for heuristic methods. For the purposes of their examination of validation and verification of heuristics and algorithms, they use the terms "heuristic" and "algorithm" interchangeably.

They define a heuristic method as a "well-defined set of steps for quickly identifying a high-quality solution for a given problem." (Barr, et al., 1995:3) Several concepts implicit in this definition are critical. First, the concept of quality, which has to be measured by the pre-defined evaluation metrics or criteria for the problem. Second, the solution is assumed to be feasible, meeting all problem constraints. Third, the speed of solution; heuristics will characteristically be applied in those situations where a *timely, good* solution is deemed more critical than an *exact* solution. Identifying these concepts is key to designing and reporting on a heuristic method.

Barr, et al., point out that, although no set standards exist for publishable algorithmic research, a heuristic method makes a contribution if it is: (Barr, et al., 1995:6)

- Fast—produces high-quality solutions more quickly than other approaches

- Accurate—identifies higher-quality solutions than other approaches

- Robust—less sensitive to differences in problem characteristics, data quality, and tuning parameters than other approaches

- Simple—easy to implement

- High-impact—solves a new or important problem faster and more accurately than other approaches

- Generalizeable—has application to a broad range of problems

- Innovative—new and creative in its own right

Additionally, a report about the heuristic's performance is valuable if it is:

- Revealing—offer insight into general heuristic design or the problem structure

- Theoretical—provide theoretical insights, such as bounds on solution quality

Ignizio (1971) made a call for establishing standards for comparing the performance of algorithms. Specifically, he wanted pertinent societies (ORSA, TIMS, SIAM, AIEE) to ordain standards with which an author must comply before *publishing* an algorithm in any of the journals. He listed the essential factors which he thought would apply to algorithms submitted for publication: (Ignizio, 1971:9-10)

1. Dates of study

2. Dates of computation

3. Computer used

4. Programming language used

5. Amount of internal storage used

6. Amount and type of external storage used

7. Storage requirements as a function of problem size

8. Total number of problems attempted

9. Total number of problems solved

10. For problems solved:

    a) Computation time range

    b) Average computation time

11. Sizes of test problems

    a) Number of constraints and variables

    b) Total problem byte requirements

12. Method of generating test problems

13. Method of validating the algorithm's accuracy (where accuracy is the difference between the optimal solution and the solution produced by the algorithm)

Ignizio suggested the following measurement standards: (Ignizio, 1971:10)

1. Computation time. Report all computation times in the same unit (e.g., seconds). Report both:

    a) Total computation time

    b) CPU time only

2. Accuracy

    a) Undefined, if the problem is not solved or does not converge and no bounds are known

      b)   Error relative to a bound, if the bound is known and the problem does

               not converge

      c)   Error relative to the optimal solution, if the problem is solved and the

               optimal solution is known

Ignizio does grant that some things cannot be standardized, such as programmer efficiency. He argues convincingly, however, that implementing standards will establish control over the controllable factors and provide a more representative result.

Recognizing the need to establish a concise set of principles for guidance, Jackson Boggs, Nash and Powell, sought to clarify existing guidelines for assessing computational tests. Their first consideration, similar to Ignizio's (1971) suggestion, was to specify the guidelines for publishing papers making claims about computational performance: (Jackson et al., 1991:414)

1. Results presented must be sufficient to justify the claims made

2. There must be sufficient detail to permit reproducibility

3. It is not the referee's job to reproduce the results

The third item is to encourage the free flow of scientific knowledge, while still encouraging the referee to ascertain the satisfaction of the first two items.

Based on the degree of the claim of computational studies, Jackson et al., classify the studies into different groups: (Jackson et al., 1991:416)

1. Preliminary testing: Those intended to show the feasibility or promise of a new algorithm, one generally not intended for widespread distribution. Demonstration of performance on several appropriate problems is sufficient.

2. More detailed experimentation: Intended to assess the strengths or weaknesses of an implementation. A range of problems should be examined, and greater detail of the implementation of the algorithm must be provided.

3. Detailed comparison of the performance: The most difficult type of experiment, this governs comparing the algorithm with prominent methods available. Performance measures must be appropriately chosen, based on the purpose for which they were designed.

Their main issue was to delineate at what point a detailed, rigorous, statistically-based computational experiment must be performed. They concluded that at the point when an author makes a "comparative" claim, specifying that one algorithm is "better" than another on some measure, he must provide enough evidence to support the claim (usually computational results). (Jackson et al., 1991:416)

Next, they addressed the concern of proprietary software. They recognized the desire of some individuals or companies to protect their intellectual property, but felt that the theoretical framework of an algorithm could be presented without disclosing the *implementation* of the algorithm. (Jackson et al., 1991:417-418)

Jackson, et al., recommended that standard test sets and test set generators be used whenever possible, to enable comparisons with results from other existing algorithms. When new problems are used in a computational evaluation, they should be submitted for free distribution, or else maintained by the author long enough to allow others to verify the experimental results. (Jackson et al., 1991:420)

They categorized the performance measures used in comparing commonly used algorithms into four groups: (Jackson et al., 1991:420-421)

1. Efficiency: A measure of both the computational effort required and the quality of the solution (CPU time, number of functional evaluations, number of iterations)

2. Robustness: Ability to recover from an improper input

3. Reliability: The size of the class of problems the code can solve

4. Ease-of-use: The amount of effort required to use the software (set-up, documentation, structure)

They furnished guidelines on analyzing the results. Experimenters should do more than report the solution times of their algorithm; if they use a test problem generator, they can examine many test problems, and can perform a statistical analysis of the results. Even if the sample size of the results is small, Jackson, et al., suggest using appropriate statistical tests to gain insights into the problem.

Finally, they emphasize documenting the type of platform upon which the program was run, particularly when using multi-processor (parallel) computers.

Sklar, Armstrong, and Samn (1990) took an interesting approach in evaluating two of their heuristics. They investigated a related airlift problem, where the objective was to minimize the number of *crews* required in the airlift, subject to crew rest requirements and the completion of all missions within the specified time frame.

They simplified the problem considerably by assuming that all the routes, missions, and associated aircraft had already been defined. Additionally, they did not consider aircraft internal configuration, ground staff or parking availability, airport

curfews, deadheading of crews or crew training. They justified their "simpler" model by claiming that it could be used to provide quick insights into an airlift exercise, and provide a *starting point* for more elaborate computer simulations which can take the other factors into account. (Sklar, et al., 1990:63-64).

They compared their heuristics two different ways. First, they compared their two heuristics against each other on several criteria: quality of solution, bounds on solution, and computational effort. They found that their Algorithm A had solutions requiring about 10% fewer crews than Algorithm B, but that often the problem size precluded the use of Algorithm A. They pointed out that Algorithm A could provide a lower bound on the number of crews required and an upper bound on how far from optimal a solution might be. (Sklar, et al., 1990:73)

Second, they compared the algorithms with a simulation approach. They used an Air Force simulation model employed to estimate the minimum number of aircrews required to achieve a target utilization rate for a fixed number of aircraft. The desired utilization rate is determined by the amount of cargo to be transported and the aircraft capacity; they report the actual *achieved* utilization rate is determined mainly by the number of aircrews available and where those crews are propositioned along the route. The Air Force policy was to use a rule of assigning a number of crews at an enroute base proportional to the number of aircraft transiting that base.

Sklar, et al., determined the optimal value of the constant of proportionality by simulation, and compared that to the value they found via their algorithms by using an actual, realistic airlift scenario for both algorithms. They declared that their algorithms in

46

general performed at least as well as and usually better than the Air Force's assignment policy. Additionally, the simulation exercise consumed several hours of computer time; their algorithms used only a few seconds. (Sklar, et al., 1990:76)

Classifying Heuristics and Algorithms

Barr, et al., classify experiments with algorithms into two categories: comparison of different algorithms for the same class of problem, and characterization of an algorithm's performance in isolation. (Barr, et al., 1995:7) In the first category, comparison, the preferred method is to test the algorithm against the best competition, usually a well-known or published heuristic or algorithm. "If other methods do not exist, then a more general method, such as one based on linear or integer programming, or a simple greedy approach, should serve as a baseline." The second category, describing the algorithm's performance, has an objective of gaining understanding of the method and the factors which influence it.

Test Sets

Some authors, such as Bodin and Golden (Bodin and Golden, 1981) have tried to classify algorithms according to their ability to solve certain types of problems. Rardin and Lin (1982) point out that investigators who wish to study algorithms in detail generally must rely on experimentation. (Rardin and Lin, 1982:9) They considered the problem of finding adequate test problems to realistically challenge an algorithm. They list three sources of test problems: published test examples conceived by researchers, documented data sets taken from "real world" applications, and randomly generated

problem sets. They tend to favor the real data sets or the randomly generated sets; they feel that the published test examples are usually small in size, specialized to demonstrate a certain particular algorithmic behavior, and that there are few examples.

One uncertainty about deliberately performing tests on small data sets is *scaling*. Not only would the researcher have to be concerned about the validity of extrapolating those results to what could be expected with realistic problem sizes, but there are the considerations of solution time or storage the algorithms would require to solve large problems. Forecasting performance would be an issue necessary to face.

Rardin and Lin (Rardin and Lin, 1982:10) point out that real data sets are generally preferred, but there are problems with them. The data may not fit the particular algorithm that the researcher wants to investigate, or if the data are drawn from a test population, there may be questions about how well they fit the actual population of interest.

Problem sets produced by random generation code are another option. For those who question why an investigator would want to test anything other than an actual sample of the subject population when doing so would raise doubts about the validity of the conclusions from the experiment, Rardin and Lin explain that it is the same reason why rats are used in medical research and models of aircraft are used in wind tunnel experiments. (Rardin and Lin, 1982:11) Experiments with laboratory models are not restrained by physical or ethical considerations. They offer a number of advantages: (Rardin and Lin, 1982:11-12)

1. Quantity. If an algorithm is specially "tuned" to a particular data set, it may behave very differently when applied to another similar problem. An experimenter with limited numbers of data sets may not detect the anomaly, but an experimenter who can run the algorithm on a large variety of randomly generated data sets may.

2. Variety. A computational researcher would be interested in testing his or her approach against problems spanning the population of interest in order to gain confidence about the results and insights into the nature of the problems that the algorithm may find easy or difficult. A cleverly-designed problem generator can produce a nearly unlimited variety of problem sets for testing.

3. Measurability. Most computational experiments are concerned with determining the solution time, the quality of the solution, or a combination of the two. When test problems are drawn from real world problem sets, often the optimal solution and value are unknown. In these cases the investigators must usually resort to comparing their results to that of the best known solution. If, on the other hand, the problem sets are generated, they can be constructed such that the optimal solution is already known, giving the experimenter a better standard against which to measure the algorithm's performance.

4. Portability. A requirement for other researchers or referees to be able to reproduce someone's efforts is that they have access to the same data set. The Internet has certainly advanced the portability and accessibility of data for researchers, but for extremely large data sets, availability can still be problematic. Additionally, the largest data sets cannot be published in standard journals. If the data set were instead

produced by a random generation scheme, the code for that would be of a more manageable size. In addition, a wide range of large data sets can be easily generated by changing the generator's parameter settings, and any other researcher with an equivalent computer and the same parameter settings should be able to reproduce the problem sets.

5. Reportability. Summary of results from testing large data sets can be easily reported in a table, tabulated by parameter settings and random number seeds used.

Rardin and Lin addressed the concern about the validity of the conclusions drawn from testing on samples drawn from a population. They remark that traditionally, researchers could only test on whatever data they had available, and hope that those data were representative. They "believe serious attention to formal validation of test problem sources could add a great deal of confidence to results without sacrificing the convenience of experimentation on laboratory models." (Rardin and Lin, 1982:12-13) They also believe that most researchers are likely to do at least one form of screening for validity: face validity. Most researchers informally check to see if they match real data, at least in important parameters like coefficient size and density.

To answer the question whether inferences about algorithm performance made by experimenting on test problems is valid for the whole population, one method would be to compare the algorithm's performance on real problems to its performance on sample test problems. A match in the performance lends credence to the conclusion that the algorithm may be acceptable. Rardin and Lin suggest a more rigorous method to measure algorithm performance, focusing on: (Rardin and Lin, 1982:13)

1. Robustness. Are the solution times and accuracies from proposed test problems comparable to those from real data?

2. Convergence Rates. Are the rates and patterns of algorithmic convergence with test problems typical of those obtained from real data?

3. Step Utilization. Is the distribution of activity within the steps and phases of the algorithm similar to that expected from real data?

4. Numerical Stability. Are controls on numerical accuracy (e.g., basis reinversion) in solving test problems required as often as, or more often than, with actual data?

5. Storage Utilization. Do the patterns of memory utilization and storage with test problems mirror those with real data?

Rardin and Lin conclude that deciding on a test problem source is a dilemma facing each computational experimenter. Until convenient techniques for validation problem generators and for forecasting the impact of problem size are developed, researchers will have to make the tradeoff between the convenience of using random problem generators and the confidence from utilizing actual data. They recommend that the tradeoff be made based on the purpose of the testing: If the testing is being conducted early in the development of an algorithm, when testing the viability of the concept of the algorithm, experimental convenience is likely to be a greater concern than the validity of the test set, so random problems are typically satisfactory. As the algorithm matures, and requires only fine tuning, validity becomes the main concern, and the experimenter should favor testing against real applications data. Finally, examiners concerned about

verifying an algorithm should stress the use of randomly generated data sets, since they would find it important to know the optimal solution and have access to a large variety of problem types.

Measuring Performance

Barr, et al., specify several questions to consider when testing an algorithm. (Barr, et al., 1995:8)

1. What is the quality of the best solution found?

2. How long does it take to determine the best solution?

3. How quickly does the algorithm find good solutions?

4. How robust is the method?

5. How "far" is the best solution from those more easily found?

6. What is the tradeoff between feasibility and solution quality?

The performance measures fall into three general areas: solution quality, computational effort, and robustness.

Solution Quality

If an algorithm which determines an optimal solution is being considered, the user is most likely interested in its speed and rate of convergence toward that optimal solution. When a heuristic is under consideration, the user is more likely concerned with how close to the optimal solution the heuristic comes. If an optimal solution is available, it can be used for a clear comparison to express a percent deviation from optimal. If an optimal

solution is not available, the comparison should be made from a tight upper (lower) bound to determine the percent deviation.

Computational Effort

Computational effort is an examination of the speed of the solution. Several measures include: (Barr, et al., 1995:10-11)

- Time to Best-Found Solution. A measurement of the time of all preprocessing, computation and postprocessing to find the "best" solution established as the baseline.

- Total Run Time. The execution time up to the algorithm's stopping criterion.

- Time Per Phase. When the heuristic is multi-phase (i.e., initial feasible solution, improved solution, final solution), a measure of time and solution quality at the end of each phase.

Any reporting method must account for differences in computational effort on different computing systems.

Another method of reporting solution effort when the result converges toward the best solution is the ratio of time to reach a solution within 5% of the best-found solution to the time to reach that best solution: (Barr, et al., 1995:11)

$$r_{0.05} = \frac{\text{time to within 5\% of best}}{\text{time to best found}} \tag{2}$$

When the heuristic converges slowly toward a solution, this provides a measure of the computational effort to reach a "good" solution, even if it is not the optimal solution.

Several authors chose to use the computational effort as their prime measure in evaluating algorithms. Braitsch (1972) did a comparison of four quadratic programming algorithms. He recognized that he needed to use some other criterion for comparison than processing time, to compensate for discrepancies due to programming or computer differences. He chose to use iteration count as the main criterion to compare the rates of convergence, but he did try to consider the number of tableau elements that must be transformed at each iteration to draw conclusions about what the accurate measure of computer time *should* be. Golden (1976) compared two shortest path algorithms and used the running time as the main rating criterion. He compared them as well on the basis of the variable list length.

Some authors recognized the need to define computational effort in terms more general than running time or iteration counts, taking into consideration the operating environment as well. Glover, et al., (1974) did an in-depth computational comparison of solution algorithms for transportation problems. First, they performed all the testing on the same computer, with the same compiler, and the same problems. They then solved the same problems with one of the algorithms on three *different* computers. They found that on a computer traditionally considered to be 3-4 times faster (than the original one used), the algorithms ran an average of only 12% faster. They used this demonstration to stress the lesson that "in any attempt to compare the performances of different codes, it is essential that the same problems, the same machine, and the same compiler be used." (Glover, et al., 1974:808-809)

54

Hooker (1995) took an innovative approach toward evaluating algorithms. He claims that we have an over-reliance on competitive testing to compare the performance of algorithms, resulting in algorithms which are specially "tuned" to perform well on the specific type of problem under consideration, wasting scientific effort in the process. He suggests instead that the development of algorithms concentrate on advancing our understanding of *what* makes an algorithm work well, in a scientific approach of controlled experimentation, rather than simply competing on the basis of speed:

> We have saddled algorithmic researchers with the burden of exhibiting faster and better algorithms in each paper, a charge more suited to software houses, while expecting them to advance our knowledge of algorithms at the same time. I believe that when researchers are relieved of this dual responsibility and freed to conduct experiments for the sake of science, research and development alike will benefit. (Hooker, 1995:41)

Robustness

Robustness is a determination of the ability of the heuristic to perform well over a wide range of test problems. Barr, et al., encourage the tester to demonstrate the robustness over the wide range of problems before fine-tuning it for a specific case or scenario. Likewise, a sensitivity analysis of the parameter settings gives a gauge of the robustness of the heuristic. They also encourage testers to report negative results when the heuristic fails a specific problem type.

Criteria

Once it has been determined *how* to test heuristics for comparisons, it still must be decided what criteria will be the measures of interest for the comparison. Golden and

Stewart recognized that there are several valid criteria for comparing heuristic algorithms: (1985:208)

    a. <u>Running time</u> is a key consideration between competing algorithms.

    b. <u>Ease of implementation</u> is another important consideration, particularly if the algorithm only marginally outperforms an alternative which is simpler to implement.

    c. <u>Flexibility</u> refers to the ability of an algorithm to handle variations in the problem.

    d. <u>Simplicity</u> in algorithms more readily permits various kinds of analysis.

Nonetheless, implicitly they agree that the most important criterion for comparison is the solution quality—how well it performs on its design measure of merit. They devote most of their effort to a discussion of methods for comparing heuristics.

Statistical Comparisons

When an experimenter interested in establishing the usefulness of a heuristic has determined what test standards to use for the testing, and which criteria are of primary interest, the actual statistical means of making the comparison must still be determined. This section will survey several of the statistical tests which can be used for the comparison.

Dominance

Perhaps the easiest way to compare heuristics is to calculate selected descriptive statistics and identify the performance of heuristics in terms of these statistics. Several of the comparisons which might be made for a minimization problem (such as when the measure of interest is run time) are: (Golden and Stewart, 1985:210)

a. The number of times a heuristic is best or tied for best

b. Average percentage above a lower bound

c. Average rank (among multiple heuristics)

d. Ratio of solution to lower bound

An algorithm which outperforms others on all of these measures may be said to dominate them.

Wilcoxon Signed Rank Test

A more systematic method of making comparisons between heuristics may be to use the Wilcoxon Signed Rank Test, a well-known non-parametric statistical test. A brief description of the Wilcoxon Signed Rank Test is provided. X and Y represent two different heuristics tested, x_i and y_i represent the performance of each respective heuristic on observation i with regard to the measure of merit of interest, E[x] and E[y] represent the expected value of the performance of the respective heuristics on the measure of merit, and i ranges from 1 to n (the number of observations):

Assumptions: (Golden and Stewart, 1985:210-212)

a. The data consist of matched pairs (x_i, y_i), with the difference defined as

$$d_i = x_i - y_i$$

b. Each d_i is a continuous random variable

c. The distribution of each d_i must be symmetric

d. The pairs (x_i, y_i) represent a random sample from a bivariate distribution

Given $E[z]$ is the expected value of some measure of interest z, and x and y are the measures for two procedures which we wish to compare, then our statistical test would develop as follows:

Null Hypothesis:

$$H_0: E[x] = E[y]$$

Alternative Hypotheses:

$$H_a: E[x] \neq E[y], E[x] > E[y], or\ E[x] < E[y]$$

Test Statistic:

The test statistic, W, is computed by ranking the absolute value of the differences $|x_i - y_i|$. Ties in the ranks are resolved by using the average of their ranks. If a difference of zero is measured, that observation is discarded, and n is decremented by one. A signed rank, R_i, is given to each observation, with magnitude determined by the rank of the observation, and sign determined by the sign of $x_i - y_i$. W is then defined by the sum over i of R_i.

Rejection Region:

The rules for rejecting the null hypothesis (at the selected α significance level) are given below. The alternative hypotheses are listed with the corresponding rejection regions, where the critical values are found in most texts on nonparametric statistics such as Conover (1980):

Table 3

Rejection Rules

$E[x] \neq E[y]$	$W > W_{1-\alpha/2}$ or $W < W_{\alpha/2}$
$E[x] > E[y]$	$W > W_{1-\alpha}$
$E[x] < E[y]$	$W < W_{\alpha}$

Golden and Stewart point out that for $n \geq 10$, the critical values can be approximated by:

$$W_\alpha = Z(\alpha)\sqrt{n(n+1)(2n+1)/6} \qquad (3)$$

Rejection of the null hypothesis implies that we are unable to accept the hypothesis that $E[x] = E[y]$; failure to reject the null means we would not make that conclusion at the given confidence level. The Wilcoxon Signed Rank Test determines if a difference exists in the location of two populations, by comparing the ranks of the differences on two heuristics at a time. It is a more powerful test than the sign test described below, but it requires stronger assumptions be made. (Golden and Stewart, 1985:212). Since it is only useful for comparing two heuristics at a time, an experimenter comparing three or more heuristics at once might consider using the Friedman Test described below.

Sign Test

A variant of the Wilcoxon Signed Rank Test is the sign test. The sign test is not as powerful a statistical test as the Wilcoxon Signed Rank Test; however, it does not require the same stringent assumptions as the Wilcoxon Signed Rank Test. (Golden and Stewart, 1985:212) If we are testing two heuristics (X and Y) and assume that they are equally likely to produce a better solution on the measure of interest, i.e., $P[y_i - x_i > 0] = 0.5$, then the probability that n or more positive (or negative, for that matter) differences would be found in m observations is:

$$\sum_{k=n}^{m} \binom{m}{k} (0.5)^k \cdot (0.5)^{m-k} \tag{4}$$

A low probability would lend credence to the supposition that the two heuristics are *not* equally likely to produce the better solution.

Friedman Test

When making comparisons of three or more heuristics, one may consider using the Friedman test rather than trying to compare combinatorial pairs of heuristics. (Golden and Stewart, 1985:212) The only assumption required by the Friedman test is that the samples from the heuristics form a normal distribution with common variance.

Let R_{ij} be the rank (from 1 to k) given to heuristic j (j = 1, ..., k) on problem i. Again, in the event of a tie, average ranks are used. This test is analogous to the ANOVA test of homogeneity and is typically used to test the null hypothesis:

Null Hypothesis:

> H_0: $E[X] = E[Y] = ... = E[Z]$ for each of j different heuristics.

Alternative Hypothesis:

H_a: At least one of the means is not equal to the others

Test Statistic:

$$T_F = \frac{(n-1) \cdot \left[B_F - nk(k+1)^2 / 4 \right]}{A_F - B_F} \tag{5}$$

where:

$$R_j = \sum_{i=1}^{n} R_{ij} \quad j = 1, \ldots, k \tag{6}$$

$$A_F = \sum_{i=1}^{n} \sum_{j=1}^{k} (R_{ij})^2 \tag{7}$$

$$B_F = \frac{1}{n} \sum_{j=1}^{k} R_j^2 \tag{8}$$

and where k is the number of heuristics under comparison.

Rejection Region:

$T_F > F_{1-\alpha, (k-1), (n-1)(k-1)}$

Rejection of the null hypothesis suggests that the researcher may want to investigate further to determine which is the better heuristic.

McNemar Test for Significance of Changes

In the event the investigator suspects that there will be a high proportion of instances when the different heuristics "tie" on their performance, one other way to test for differences is the McNemar Test for Significance of Changes (Conover, 1980:130-133). This test presupposes that the data are not ordinal as in the sign test, but nominal, with categories "0" and "1". The pair (1, 0) indicates X_i outperforms Y_i on the measure of merit; (0, 1) indicates Y_i outperforms X_i. McNemar's test is a variant of the sign test described above, subject to the following assumptions:

1. The pairs (X_i, Y_i) are mutually independent.

2. The measurement scale is nominal with two categories for all X_i and Y_i.

3. The difference $P[X_i =0, Y_i =1] - P[X_i =1, Y_i =0]$ is negative for all i, or zero for all i, or positive for all i.

Null Hypothesis:

H_0: $P(X_i = 0, Y_i = 1) = P(X_i = 1, Y_i = 0)$ for all i

Alternative Hypotheses:

H_a: $P(X_i = 0, Y_i = 1) \neq P(X_i = 1, Y_i = 0)$ for all i

These hypothesis can simplify into:

Null Hypothesis:

H_0: $P(X_i = 1) = P(Y_i = 1)$ for all i

Alternative Hypotheses:

H_a: $P(X_i = 1) \neq P(Y_i = 1)$ for all i

Our null hypothesis is that we expect the number of (1,0) and (0,1) pairs to be equal. The test statistic and rejection rule both depend on the sample size, and their calculation is fairly complicated. The interested reader may refer to a nonparametric statistics text for details. (Conover, 1980:130-133)

For this test, rejection of the null hypothesis would suggest that two heuristics do *not* perform equally well. Use of a one-sided rejection rule can determine which heuristic provides the best solution.

Expected Utility

Each of the tests described above are reasonable statistical tests for comparing heuristics, though they are all tests of *location* (i.e., mean or median), and give no indication of the *shape* of the distribution. In consideration of this, Golden and Stewart proposed an expected utility approach to compare multiple heuristics. They explain that, though it may be a simple and useful approach, it depends on somewhat arbitrary assumptions. (Golden and Stewart, 1985:213)

The focus of the technique is the notion that the desired heuristic not only performs well on average, but also rarely performs poorly. Thus, the expected utility approach reveals a consideration of the downside risk as well as the expected accuracy.

Step 1. Fit a gamma distribution to the histogram of frequency vs. performance deviation from a lower bound.

Step 2. Select a risk-averse decreasing utility function of the form $u(x) = \alpha - \beta e^{tx}$, where $\alpha, \beta, t > 0$ and x is the percent deviation from the lower bound.

Step 3. Calculate the expected utility for each heuristic and select the one that

yields the largest value.

This procedure is described in detail in Golden and Stewart. (Golden and Stewart, 1985:213-214) In step one, they chose a gamma function for its computational convenience (it has a simple density distribution, and parameter estimation via the method of moments is trivial). In step two, they chose a utility function with t as a measure of the risk aversion, though they give no rigorous justification for their choice of a risk averse utility function.

Reporting

Once the testing is completed, the researcher must report the results in a rigorous manner. These guidelines are established to help authors report the computational testing results to others:

1. Reproducibility—document to allow others to substantially reproduce the results. (Barr, et al., 1995:19)

2. Computing Environment—document the make and model of the computer; number, type and speed of processors; operating system and version; compiler settings; and system load. (Barr, et al., 1995:19-20)

3. Timing—document how times are measured, differentiating between user, system, and real time. (Barr, et al., 1995:20-21)

4. Parameter Selection—if the code allows for different parameters, specify the parameter settings, the process by which they were selected, reason for changes in parameters for different problems,

evidence that the parameter values are generalizeable, and estimates of time to fine-tune the algorithm. (Barr, et al., 1995:21-22)

5. Statistical Techniques—when the main thrust of the research is to demonstrate that a particular heuristic outperforms another, a statistical comparison of results should be included. If sample size is large enough, t-tests or ANOVA may be used. If the sample size is not large enough, nonparametric techniques (e.g. sign test) may be used, even if the intent is to show that there is no statistical difference between the solutions. (Barr, et al., 1995:22)

6. Variability of Results—experiments should be designed to reduce variability by running longer experiments and using more data. If the experiment is done on a computer which may be shared, it should be run on a lightly loaded machine. (Barr, et al., 1995:23)

7. Analysis and Interpretation—report not only the experimental results, but the investigators conclusions. Detail the exploration of quality vs. computational effort, time vs. problem size, and robustness vs. quality. Unexpected or anomalous results should be highlighted, and explained when possible. (Barr, et al., 1995:23)

Summary and Applicability

The principles of validation and verification apply most directly to models which attempt to accurately represent some aspect of the real world. Simulation and wargaming are prime examples of those types of models. However, many of the same techniques

used to establish those models' validation and verification, as described above, can be applied in testing the credibility and applicability of the DAKOTA scheduling program, or any heuristic method.

A comment common to most of the literature reviewed was an insistence that the single most important criterion in the evaluation of any heuristic is its effectiveness in meeting the requirements *of a given problem.* With this point as the main concern, this thesis delineates how to evaluate DAKOTA for the specific problem of its application to USAFE's OSA scheduling problem.

Obviously, this is not an all-encompassing review of solution efforts. For problems ranging from the Traveling Salesman Problem, to Vehicle Routing Problems, to Job Shop Scheduling, as the problem size increases, the difficulty of solving the problem *to the optimal* explodes. Thus, nearly all the authors cited recommend using a heuristic, or approximate, approach to the solution. The question then becomes: If we cannot guarantee that our solution to the Vehicle Routing Problem is optimal, how good is it? That issue is addressed in subsequent chapters. But first, in order to gain an understanding of the environment in which DAKOTA works, the next section compares several passenger airlift scheduling programs in use today.

III. Discussion Of Contemporary Scheduling Systems

Alternative Scheduling Systems

The problem of scheduling OSA flights is not unique to USAFE. Different commands, services, corporations and even nations face the daunting task of developing schedules to best utilize their fleets and personnel. This chapter examines several of the scheduling systems which are used by agencies today, compares them and their features, and reviews the opinions of several subject experts regarding the systems. The systems considered are: Andromeda, ITAS, SAMMS, and JLIS. Andromeda is discussed in depth, since it is the system currently used by USAFE.

Andromeda

Currently, USAFE schedulers use the Andromeda FS flight planning system to accomplish their work. The Andromeda system, by Camp Systems, Inc., is a mission management system which assists in both scheduling flights and in tracking the flights in progress. It is a popular planning tool in the corporate world, used by 78 organizations and customers including the Federal Aviation Administration (FAA), Canada's defense department, Norway, Sweden, and the United Kingdom's Royal Air Force's Queensflight.

Andromeda FS is eleven years old, so it has most of the "bugs" worked out of it. (Finn, 1996) Most of the customers use a DOS-based version, though several users have

a Windows-based version. The most recent release is a Windows 95 version. It is network-compatible, so schedulers can connect and do their work from remote locations.

Andromeda is not an automated scheduling or decision support system. It provides a convenient, organized method of coordinating the needs of the passengers, the crews' training and scheduling requirements, and the aircraft and airfield-related constraints.

When the schedulers need to schedule a request, they look at missions and requests already scheduled, manually determine the mission on which to schedule the request, and then enter that information into the Andromeda system. Andromeda then calculates the required departure and arrival times, distances, fuel requirements, and other flight data for the mission.

Andromeda does not automatically select the mission aircraft, but it does simplify many of the tasks associated with scheduling, like computing enroute times. If a nonaligned nation must be circumnavigated, it is easy to change the required flight time manually. For instance, Andromeda may allot 1.3 hours for a flight from Ramstein AB to Aviano AB, Italy. The schedulers can manually change the time to 2.0 hours to account for flying around Austria, if necessary.

Table 4, on the following page, lists many of the features of Andromeda FS:

TABLE 4

ANDROMEDA'S FEATURES

FEATURE	DESCRIPTION
Airport Atlas	
Airports	Airports, location, identifiers
Airport Facilities	Services (fuel, etc.) available
Tracks Slot Reqts	Flight planning system accounts for "slot times" for takeoff or landing required at some airports
Runway Length	Length of longest runway available
Sunrise/Sunset	Gives seasonal changes in sunset and sunrise
GMT Conversion	Gives times in universal GMT/"Zulu" and local
Curfews	Quiet hours at appropriate airfields
Jeppesen Data	Includes navigational aids and instrument approaches available
Aircraft Data	
Speed	Accounts for airspeeds of different types of aircraft
Fuel Endurance	Accounts for fuel ranges of different types of aircraft
Variable Burn Rate	Accounts for changes in fuel consumption rate with changes in temperature and aircraft altitude
A/C Limitations	Aircraft weight, fuel, runway needs, seats available, other limits
Aircrew	
Crew Scheduling	Automatically schedules crews for flights
Crew Training	Schedules required training for crews
Crew Currency	Tracks currency requirements for crews
Crew Time Off	Tracks post-flight time off for crews
Crew Duty Day	Tracks maximum duty day for crews
Crew Logbooks	Maintains logbook of crew flight activity
Flight Mgt	
Operations Manager	Tracks fleet activity at a glance
Track Enroute A/C	Monitors status of enroute aircraft
Flight Time	Tracks total flight time on each aircraft
Flight Mgt Control	Tracks aircraft, passengers on each flight
Passenger Info	Schedules and tracks passengers, general remarks allowed
Track Schedule Changes	Audit trail of all changes made to a trip
Scheduling Checklist	Provides guidance to scheduler of scheduling requirements
Recurring Trips	Detects common flights for automatic scheduling
Radius Search	Notifies scheduler of nearby airports for refueling when needed
Flight Planning	
Seasonal Wind Factors	Accounts for seasonal winds when computing enroute times
Great Circle Routes	Determines great circle route distances between points
General	
On Line Help	Help available on-line
Reporting System	System of generating reports for flight management
Customized Reports	Supports user-defined reports
Greaseboard/Gantt Charts	Visual or Gantt chart scheduling aid
"Street Light" Warnings	Red indicates "hard" error/violation; Yellow indicates "soft"; Green is OK

In summary, Andromeda's strengths are its ubiquity, its extensive airport atlas, and its notable ease of use. Its weaknesses are the lack of a decision support system and an optimization routine.

ITAS

The In-Theater Airlift Scheduler (ITAS) was developed in 1994 for the Pacific Air Forces (PACAF). Developed through an affiliation between the Rome Laboratories and the Kestrel Institute, ITAS is under continuous development. It is not used for routine daily scheduling, most of which is currently done manually. Its purpose is to support contingency operations, such as the Hurricane Iniki evacuation or exercises (Lemmer, 1996). Designed for that aim, it runs on an Apple Powerbook notebook computer, so it is extremely mobile, a crucial feature for contingency operations.

Users input the number and types of aircraft, the number of crews, the movement requirements and port features. ITAS produces a candidate schedule, simultaneously scheduling the aircraft, aircrews, ground crews for unloading, and parking spaces at the ports. Most of the process is automated, with only a few high-level decisions made by the scheduler. The output does not necessarily represent a feasible schedule, however, so the scheduler must review the schedule, note any discrepancies (constraint violations), and input the necessary revisions for the next iteration. While this seems to be an inconvenience, the users are not vocal in their complaints; the revisions take only about five minutes, instead of the hours required before implementation of ITAS. (Burgess, 1996)

Generally, the resultant schedule is not produced in an easily readable format. One of the more convenient formats for an output is in a Gantt-like bar chart; another is in the format of an Air Tasking Order (ATO).

ITAS is extremely good at scheduling Time-Phased Force and Deployment Data (TPFDD). The TPFDDs are a relational database scheme which the U.S. Transportation Command and the component services use to specify the transportation requirements of an operation such as Bosnian airlift or hurricane evacuation. Smith and Parra (1995) report that ITAS' algorithm is extremely fast compared to other TPFDD scheduling systems. A typical TPFDD of about 10,000 movement records scheduled via alternative programs may take several hours to solve using JFAST, or 36 hours using FLOGEN. It can be solved by ITAS in *one to three minutes*. He says that it "is orders of magnitude faster than any other TPFDD scheduler known to us." (Smith, et al., 1995:6-7)

SAMMS

The 89th Airlift Wing, at Andrews AFB, MD, provides the presidential airlift and airlift for other prominent government officials. (Manney, 1996) The presidential airlift on Air Force One is handled on a case-by-case basis, due both to the visibility of the mission, and to the fact that there are only two aircraft for the mission and only one is apt to be tasked at a time.

The scheduling for the rest of their OSA is conducted with the assistance of SAMMS (Special Airlift Mission Management System), a computer program which was developed in-house for scheduling support. While it is a manual scheduling system, it does have features to aid the scheduler. It weighs predominant winds in its distance/time

calculations. It produces reports which they send to Command Post and the Passenger

Terminal to help coordinate the missions.

Their scheduling policy is to use the smallest, most cost effective aircraft possible

to support a mission. Of their seven C-20's, three C-9's, and five C-137's, an average of

five aircraft are in use each day. Most of their missions, about 75%, are flights which

will remain off-station overnight.

Maj Manney was not convinced that "optimization", in terms of reducing mileage

while serving as many passengers as possible, would be germane to their mission for

several reasons: 1. Most of their missions are RONs, where the crew and plane wait at

the airfield until the passengers are ready to return. 2. For obvious reasons, the plane will

not leave high-ranking passengers at one airport in order to travel to another to pick up

additional passengers. 3. Again, for obvious reasons, the planes are likely to travel

straight from origin to destination and back, rather than diverting to other airfields to pick

up additional passengers. Maj Manney concluded that, though an optimization

subroutine in their scheduling system may not necessarily help them, it probably

wouldn't hurt much either.

JLIS

The Joint Logistics Airlift System (JLIS) is a follow-on to the earlier Navy Air

Logistics Airlift System (NALIS). Developed in-house by the Navy Air Liaison Office

(NALO), it is the ascendant scheduling system for continental U.S. (CONUS) based OSA

missions. The NALO (Steedley, 1996) says that by direction of the Office of the

Secretary of Defense (OSD), JLIS was ordered operational as of 1 Oct 95. Not all of the bugs were fixed by the release date, but they claim significant progress since then.

JLIS has many strengths and equally severe weaknesses. Its strengths generally are derived from the versatility of the Oracle database system on which it runs; the weaknesses stem from the poor user interface. (Noble, 1996)

Strengths:

1. Users report a high degree of confidence in and satisfaction with the Oracle database. Among other things, its structure permits multiple schedulers to work in parallel without conflicts. (Steedley, 1996)

2. Security. Individuals in the units having access to the program each have their own login identification for security. (Steedley, 1996)

3. Constraints. The versatility of the database allows it to track a large number of constraints in the background, ensuring their satisfaction, flagging the operator only when there is a violation. Among the constraints modeled are aircrew restrictions (length of duty day, interval between duty periods), fuel and aircraft capacity constraints, passenger/cargo mix and available-to-load times, airfield restrictions (Notices to Airmen, quiet hours, sunrise/sunset), aircraft speed, and quarterly seasonal winds aloft. NALO reports that JLIS does a more comprehensive check on the constraints than does Andromeda. (Pellissier, 1996)

4. Automatic loading. Eases the workload of the schedulers by relieving them of the requirement to manually enter missions and requests. Each flying

73

squadron has access to the database, and they are the ones who directly input the aircraft status, maintenance schedules and periodic inspections, aircraft configuration, and crew training requirements. Additionally, the users are the ones who input the requests. Requests can be entered by standard message traffic; JLIS reads the message, automatically entering the request into the database or else flagging the scheduler with the message's date/time group for hand-entry if there is a formatting error. Secondarily, requests can be entered either by modem or via Internet, again with security-restricted access. Finally, requesters are able to hand-walk their requests through the system for manual entry. In any case, having the units or individuals enter the aircraft missions and requests not only reduces the workload on the schedulers, but ensures that the most up-to-date information is being distributed, since those with most direct access to the information are the ones who enter it into the database. (Steedley, 1996)

5. Automatic schedule distribution. Another attraction of JLIS is its automated schedule notification system. When a schedule is crystallized, JLIS automatically publishes the schedule, notifies the flying unit of the schedule, sends flight advisories to each origination or destination airfield affected, and even notifies the passengers' commanders. (Steedley, 1996)

6. Rig changes. "Rigging" is the navy's term for the aircraft configuration. For example, a C-9 can *either* carry 90 passengers *or* 45 passengers plus 6,500 pounds of cargo, depending on its rigging. JLIS will attempt to use the

current configuration for the aircraft, but if requirements dictate that the rigging be changed, the flying units are notified in time to effect the change. (Steedley, 1996)

7. DSS tool with human interface. JLIS attempts to aggregate all the flights in the database possible, but the "man-in-the-loop" system allows the scheduler to make the decision. After scheduling the high-priority requests, it attempts to aggregate the lower-priority requests whenever possible, in order to support as many as possible. When it becomes necessary to "bump" passengers or cargo, JLIS offers up the lowest-priority requests for elimination. (Pellissier, 1996)

8. The decision support system simultaneously does two things to optimize the schedule: The Evaluation of Flight Schedule prompts the scheduler to move loads to different aircraft or to add legs/stops to a scheduled flight if it would free another airplane to fulfill other requests, and the Search for Solution of Unsatisfied Requests prompts the scheduler with additional changes to help fill other unsatisfied requests. (Pellissier, 1996)

9. NALIS had been an automated system, but it made some bad assignments while following general rules and priorities. The users were not comfortable with that feature, and none of the services wanted an automatic system, so that innovation was discontinued in the subsequent system. (Pellissier, 1996)

10. Speed. According to NALO (Pellissier, 1996), the UNIX-based system has "real good" speed, responding to most inputs within five seconds, and solving most of their complex problems within about five minutes.

11. Flexibility for carrier support. Much of the impetus for naval aviation airlift is to provide aircraft carrier support. Since that support is focused on the departures and arrivals of the ships, whenever the carrier has to change its schedule, even by one day, the airlift "house of cards" collapses and needs complete revision. JLIS' database structure is designed to reduce the impact of mission changes. (Pellissier, 1996)

Weaknesses:

1. Irrespective of potential future releases, the current version of JLIS is *universally* condemned by all users for its lack of user-friendly features. There is no Graphical User Interface, and no hot keys. It is a DOS-based program, with no mouse support, so it requires repeated "tabbing" to get to different fields. There is no point and click or drag and drop capability. (Noble, 1996)

2. Due to its complexity, even those who developed JLIS are concerned that it has a very steep learning curve, often taking weeks to learn. They assert that, once users become more familiar with the program, they are less intimidated by it, but that it is still a complex program to use. (Pellissier, 1996)

3. Another user (Noble, 1996) expressed reservations about the "mindset" required by the system: He said that it is written from an "aircraft driven"

perspective, instead of the customer-service "request oriented" approach that the Air Mobility Command uses. He also found that due to the accounting system in Oracle, they cannot simply delete a record, as they would like to do with a request if it is canceled.

Summary

After considering several of the scheduling programs available today, the reader can see that each program has its strengths and weaknesses. Andromeda is praised for its user friendliness and its extensive database designed to simplify scheduling and mission management. ITAS is useful for its mobility and for its ability to consider the theater airlift cargo mission. SAMMS is a useful, very specialized program for the presidential airlift mission. Its users appreciate its customized reports for schedule coordination and mission tracking. JLIS is the most elaborate and complicated system considered, with its powerful database and automatic database entry capability, but it is an extremely complex program to learn.

The next chapter discusses in detail how to evaluate the effectiveness and appropriateness of the heuristic contained within the primary program of interest: DAKOTA.

IV. METHODOLOGY AND TEST PLAN

The preceding chapter introduced several passenger airlift scheduling programs; Chapter II reviewed the current literature and the various authors' views on evaluating models in general, or on evaluating heuristics and algorithms specifically. This chapter builds on these thoughts and ideas to develop a methodology which one can use to evaluate a deterministic passenger airlift scheduling heuristic, using DAKOTA as a case study.

First, the user must determine the objective of the heuristic. This may sound fairly straightforward, but there are a number of aspects to the problem. What exactly is the objective that the "customer" wishes to achieve:

1. Is the intent to <u>minimize the amount of flight time required</u> to support the requests?

Ostensibly, yes. Saving flight time translates into saving money, which is clearly of interest.

On the other hand, currently USAFE is not under pressure to save flight time. They are authorized enough time to meet their mission of providing Distinguished Visitor (DV) airlift support. In fact, they had extra flight time remaining at the end of Fiscal Year 1995. (Maher, 1996)

2. Is the intent to <u>reduce the number of crews required</u> to perform the mission?

For the short run, yes. USAFE faces a crew shortage with the impending summer rotation cycle for their personnel. In the long run, the trend of force structuring in Europe

(and throughout the Department of Defense) has been toward CONUS basing and downsizing. Finally, the pilot shortage promises to be an issue which will once again come to the fore. Reductions in the number of crews required could become a potential concern.

There are limits, of course, in the application of any solutions obtained. The Air Force has prided itself on its quality of life and respect for families. A solution which severely limits the amount of time crews spend at their home bases and with their families may cause long-term problems. Conversely, a solution which emphasizes "Out and Backs" would be supportive of the Air Force's quality of life goals.

3. Is the system needed to reduce the number of aircraft required?

In recent years, USAFE has restructured its OSA fleet. All of USAFE-assigned OSA helicopters, C-12s and C-135s have been reassigned, as has one of the C-20s. Smaller, more cost-efficient C-21s have been gained. Undoubtedly, the restructuring is not complete; future reductions may yet occur.

4. Is the objective to reduce the number of time window violations?

This is one of the options available under the DAKOTA scheduling system. Obviously, for DV support, on-time pickups and deliveries are a concern. Ultimately, with a given set of missions available to fulfill a set of requests with time constraints, there will be a limited number of feasible solutions (if any).

The nature of the problem then becomes one of choosing whether it is more important to: 1) serve a greater number of requests, or 2) meet the time constraints of a

fewer number of requests. Lacking some automated or knowledge-based system of making those decisions, the decision is made by the scheduler.

For DV scheduling, the determination is made on the basis of the priorities (often rank) of the personnel involved. For the 89th Wing's presidential support, for instance, support of the primary customer is the only imperative; supporting other requests is usually not even considered.

5. <u>What is the objective of OSA?</u>

Ironically, one of the express objectives of OSA has nothing directly to do with furnishing support for DVs: Providing "seasoning" for inexperienced pilots (Petry, 1996). Reductions in the amount of flight time would run contrary to this objective. Besides, if the crews *are* relatively inexperienced, they may benefit more from a series of shorter flights which originate and terminate at the home base than from a scheduling system which aggregates requests and requires them to fly to twelve different destinations in six different countries over a three day span.

6. Is the objective to simplify the scheduling process in order to <u>reduce the workload on the schedulers</u>?

Initially, that was the concern posed. However, extensive interviews with the schedulers themselves (Velasquez, Maher, Mustin, Noble, Burgess, Chaffin, Goetz, Manney, 1996), especially those overseas, reveals that assigning requests to missions is a minor part of their duties; a major part of their task is to arrange the coordination between different units, different bases, and often different nations (diplomatic clearances) for the flights. Generally, they would be interested in a scheduling system not so much for its

ability to provide support for a greater number of requests with limited resources, but rather for its ability to *reduce their workload.* From their perspective, the primary concern for a system would be its flexibility, its user-friendliness, and its simplicity to learn (including the extent of its documentation). Their secondary concern would be how well it eases the manual tasks required. An example of this might be automating much of the coordination process between different agencies.

7. Is the cost of the new system the main concern?

Obviously not, since there will be costs associated with implementing any new system, but any system under consideration must be shown to be cost-effective.

8. Do they need it? Or want it?

These may be the key questions when considering the objective of the proposed scheduling system. Who proposed the development of the new system? In the case of DAKOTA, it was not the end users, the schedulers themselves, but rather Operations Analysis concerned for improving the efficiency of the process. Currently, the schedulers use a commercial system used by many Fortune 500 companies and other government agencies. While that system may not produce *optimal* schedules, measured in terms of reducing the number of flight hours required for the desired level of support, the schedulers are convinced that it meets their needs. (Lopez-Velasquez, 1996) They praise it for its efficiency and ease of use. Even *if* DAKOTA could be shown to provide savings in terms of the flight time required, barring a mandate from a higher authority, the customers of the process (the schedulers themselves) would be hard to convince that it is a system they should embrace.

9. Is the intent to develop a <u>scheduling system for future applications</u>?

This may turn out to be the key factor. For the *current* needs of the OSA scheduling mission in USAFE, the users (schedulers) are unconvinced of the need for the DAKOTA scheduling system. They believe they are able to manually schedule the missions adequately. However, if the regime of interest expands, the problem may change considerably: If the scheduler is concerned about *efficiently* scheduling up to 100 AMC requests each day, instead of 6 or 7 for USAFE, then the scope of the problem changes drastically, to the point where a human scheduler may be unable to determine an optimal solution manually. Likewise, if the scheduler is tasked with *efficiently* scheduling all of the theater (cargo) airlift for the USAFE theater, it is doubtful that manual scheduling can produce an optimal schedule.

In summary, the *purpose* of developing or implementing an automated scheduling system is not clear. Objectives can range from reducing flight time required, to reducing the number of minor constraint violations, to reducing workload on the schedulers themselves. The decision ultimately will rest with the OSA mission's operational commander.

After consulting with HQ USAFE/DON and HQ USAFE/AOS/AOOMS

(Wilkinson, 1995; Lopez-Velasquez, 1996) to determine what USAFE's objectives for an

automated scheduling system are, the objectives are classified into three general areas:

1. **Solution Quality**. Flight time, number of aircraft or crews required, number

   of soft time window violations, and number of infeasible solutions.

2. **Speed or Effort**. Scheduler workload and number of iterations.

3. **Robustness**. Brittleness and elasticity.

These three areas are discussed in the section which follows, and then a plan for testing

DAKOTA to determine how well it performs on those objectives is introduced.

1. **Solution Quality**—identifies higher-quality solutions than other approaches:

Since the USAFE schedulers are content with the system currently in use, it

would seem the greatest justification for changing to a new scheduling system would be

if it provided "higher quality" solutions than the current solution technique.

Flight Time. The most obvious test of solution quality is a measure of the flight

time required to support a given number of requests. Alternatively, the *number* of

requests able to be scheduled onto a given mission would be another measure of quality.

Walker attempted to make historical comparisons with past USAFE schedules

using real world data. Though he did not have access to historical requests, he did use

historical schedules for testing. From them he fabricated "fictional" requests upon which

to test, varying the time windows, but a more realistic test would be to use the actual

requests (see Appendix D).

Number of Crews/Aircraft Required. If the missions are defined first, and then the requests are scheduled onto those missions, DAKOTA will attempt to schedule as many of the requests as possible, while minimizing the flight time needed to accomplish them. On the other hand, the scheduler could start by defining the requests in DAKOTA, and then adding *missions* incrementally, rescheduling after each mission is added. Any requests which are insupportable by the current number of missions will be flagged in red as unscheduled. The scheduler can continue adding missions until all requests are scheduled.

Number of Soft Time Window Violations. Another useful measure of solution quality would be the number of times the heuristic violates the soft time window constraints. DAKOTA can generate multiple candidate solutions, allowing the user to select the most favorable one. An additional useful feature of the program is its "stop light" color coding scheme; soft time window violations are flagged in yellow, advising the operator of the violation and allowing them to contact the passenger to confirm the acceptability of the violation, or to propose alternate travel arrangements. The difficulty with this measure is that during manual scheduling the scheduler follows a similar process, so during the testing process the evaluator must make sure to use the *original* travel requests, not the modified requests after scheduling.

Number of Infeasible Solutions. Again, DAKOTA's coloring scheme will signal an infeasible solution in red. For a given problem set, the number of legs in the solution which cannot feasibly be flown can be compared to the alternative's solution. This is

simply a variant of the solution quality measurement described earlier, counting the number of legs supported by a given mission.

2. **Speed or Effort**—produces high-quality solutions more quickly than other approaches:

Number of Iterations to Reach Feasibility. The test above proposed noting legs with violations, contacting the passengers to confirm the acceptability of alternative travel arrangements, and then running the heuristic again. The number of iterations required would be a useful measure of solution quality, especially against a competing heuristic such as ITAS which is *designed* to be an iterative process.

The heuristic under consideration, in this case SchedGen, should be rated on how quickly it finds solutions. Below are several examples of the tests Walker used in his assessment:

Computational Effort. Walker did considerable testing on this aspect of the SchedGen heuristic. One test he conducted was scheduling nine requests onto one mission, where one of the requests was insupportable (it violated the capacity constraints). By varying *when* the insupportable request was considered, he was able to measure the number of columns generated as a function of the placement of that leg. He determined that the earlier that leg was tried and rejected, the greater the computational savings. This provided his justification for his variable ordering strategy (see Chapter II) which encourages early consideration of those requests likely to be found difficult to schedule. (Walker, 1994:103-105)

He also calculated the size of the event list as a function of the number of legs with soft time window bounds. He found that over the range from six to 14 legs, the event list size showed nearly linear growth. He repeated the test again, using hard time bounds for the pickups with nearly identical results. (Walker, 1994:105-107)

Despite the consistency of the event list size, Walker found that the computation times varied considerably. When considering the unconstrained requests (those without time windows), doubling the number of legs from six to 12 caused the CPU time to increase 166-fold. In the constrained case (requests which had time windows), the increase was 105-fold. (Walker, 1994:107)

Walker described one scenario where his heuristic failed to converge to a solution within 45 minutes, and he recognized the need for the heuristic to give feedback on solution progress and to allow the user to "back out gracefully" from similar situations. (Walker, 1994:117-118)

Next, Walker ranked the variable ordering strategies according to their CPU time. He found that the Earliest Pickup First (EPF) strategy was the fastest (averaging about 40% faster than the slowest method), but that regardless of which ordering strategy was used, the flight times required to support the same number of legs were within five minutes of each other. (Walker, 1994:108-110)

Scheduler Workload. As related in Chapter II, Ignizio asserts that computation times should be reported not just in terms of CPU time, but total time as well. Walker makes no effort to explain some measure of the *total* time required, from the pre-processing, to the post-processing, or even to include the user's input and output time. If

the algorithm must be run very frequently, or if it must be run iteratively to improve

solutions, CPU time becomes significant; for a scheduling program which might be run

weekly, or at most once or twice a day, even several minutes of processing time may not

be significant to the user. A better measure would obviously be the *user* time required to

get a workable solution. This should involve the amount of time required to run the

program, to input the data (passengers, requests, missions and aircraft), and to output the

solution in a format useful to the scheduler (for instance, TPFDD or ATO formats).

This author's experiences showed that the user time can become uncomfortably

long. Testing was performed on a limited sample during January through March 1996 on

a Sun-10, operating under Solaris 2.3. Though it should not need to be accomplished

very often, starting the database took approximately 15 minutes. The map function was

not painfully slow, even though it was fairly graphically intensive. One of the slowest

functions was in entering the data: each time the database had to be accessed there was a

noticeable delay. In many of the windows, the user must then use the mouse to move

from field to field, necessitating moving frequently between keyboard and mouse.

3. **Robustness**—sensitivity to differences in problem characteristics, data quality,

and tuning parameters.

According to Rardin and Lin (Rardin and Lin, 1982:13), robustness can be

established by determining if the solution times and accuracies from proposed test

problems are comparable to those from real data. Solving smaller historical

problems can help establish the face validity for the heuristic; solving larger

problems will lend credibility to the model's robustness.

Brittleness. Brittleness is a measure of vulnerability to errors or even small changes in the schedule. Two good examples of a brittle schedule is one which is so "tight" that it cannot tolerate any delays in a mission, or one which fails badly when there is a mission cancellation. Walker demonstrated DAKOTA's robustness by performing two experiments: He "unscheduled" an historical schedule and applied his heuristic to the same set of missions and requests. His resultant schedule reduced the resources required by two aircraft and nine flight hours (13%). Then he successfully unscheduled and rescheduled a mission already in progress as a simulation of a mission failure, demonstrating the responsiveness of his algorithm. (Walker, 1994:122-125)

Elasticity. We saw earlier that computational effort increases when the time windows are less restrictive. Elasticity is a measure of how minor changes in the time windows affects solution quality. What is the impact of changing a hard time to a soft time, or vice versa? What happens when the acceptable arrival time window increases in size?

Summary

An outline of a specific plan for testing DAKOTA is given in Appendix E. The test plan is based upon comparing the system's performance to a manual schedule. The measures of merit tested are:

1. **Solution Quality**. Flight time, number of aircraft or crews required, number of soft time window violations, and number of infeasible solutions.

2. **Speed or Effort**. Scheduler workload and number of iterations.

3. **Robustness**. Brittleness and elasticity.

The three statistical tests used in the comparison are:

1. Wilcoxon Signed Rank Test.

2. McNemar Test for Significance of Changes.

3. Sign Test.

As the DAKOTA system could not be made fully operational at AFIT, we were unable to execute any portion of the test plan outline in Appendix E or the alternative tests that were to be conducted. Appendices B and C list the problems encountered and actions taken attempting to make DAKOTA operational on AFIT's system.

Our inability to make DAKOTA operational does not, however, reduce the need to consider optimization in mission scheduling within the Air Force.

V. CONCLUSIONS AND RECOMMENDATIONS

Conclusions

Fiscal constraints will continue to provide pressure for more efficient use of resources, including Operational Support Airlift. Any means of reducing costs are likely to be eagerly accepted, particularly if they do not negatively impact accomplishment of the mission. A more efficient computer scheduling program which promises to save flight time (and hence, dollars) is one good example. However, since the implementation of a new program itself costs money, it is important that an effort be made to verify the claims of potential savings before rushing headlong into a new system.

The DAKOTA scheduling system has been proposed for use in USAFE's Operational Support Airlift scheduling. This thesis has examined the OSA scheduling topic and concluded that exact methods are intractable due to problem size. Consequently, heuristic methods must be considered. However, the issue then arises of how we determine the effectiveness of a heuristic, or even, how we decide what to use for the standards by which we judge the heuristic. Is an exact solution important if it takes too long to find? Is a timely, but "good", solution more important? What is meant by "good"?

This thesis contemplated the concepts of validation, verification and accreditation as they apply to heuristic methods. It examined the measures of performance which should be considered when evaluating a heuristic or an algorithm, and the statistical tests

which might be used to distinguish how well it performs. Finally, it provided a detailed test plan for a heuristic, using DAKOTA as a case study.

Recommendations for Future Research

Several areas remain open for potential future research:

1. Implementation of the test plan on the DAKOTA program may yield promising results.

2. Installing the DAKOTA program needs to be simplified. Either the system requirements for the program to run properly need to be defined, or a thorough, well-documented installation routine needs to be developed. A listing of the program's files and directories and their purposes would be helpful, as would a list of other required files which are not included with the program.

3. DAKOTA could be modeled to respect any of a number of other constraints, from crew duty limitations, to airfield capacity and time limitations, to aircraft performance data, to cargo theater airlift models considerations.

4. A "portable" optimization routine could be identified for use within any of the multiple existing scheduling systems. Incorporating that optimization routine into other scheduling systems could improve scheduling efficiencies, and potentially enhance standardization between systems.

5. Currently, the human scheduler must define the timeframe for missions for the DAKOTA program. Since the specifics of the mission definitions affects the

feasibility of supporting requests, an enhancement to the program which "suggests" to the user efficient mission definitions would be constructive.

6. Since most heuristics have multiple objectives, a decision analysis framework could prove useful in determining appropriate weights for each of the objectives and in judging the effectiveness of the heuristic.

7. To assist in testing and comparing scheduling system heuristics, a documented data set taken from real world airlift scheduling applications should be developed for use by the Air Force and Department of Defense.

8. The elasticity feature described in the test plan phase might prove useful: It can be used to determine how much flight time (and money) could be saved by encouraging the passengers to be more flexible in their requested travel times. If each passenger can be induced to provide a two (or even three) hour time window for both departure and arrival, how much flight time would be saved?

9. Clearly, for small problems, even the manually-produced schedule may be an optimal solution. Further study should be made to determine at what point the manually-produced schedule varies materially from an optimal solution. That is, for what size problem does an automated optimization routine become necessary?

10. Finally, and probably most importantly, if DAKOTA can be shown to save flight time, an estimate of the amount of flight time (and money) which would

be saved over a year could provide a very convincing argument for adopting an automated optimizing scheduling system.

Heuristics and algorithms, the engines of optimization, have a key place in today's Air Force. With restrictive budgetary limitations, the need to do more with less will increase. Developing flexible, user-friendly systems and training personnel to use them will be essential to meeting the Air Force's mission, both today and in the future.

Appendices

Appendix A: Walker's Tests of DAKOTA

In his dissertation, Walker (1994) performed a number of tests on his algorithm and program. Several of his tests, and some of his conclusions, are summarized below:

1. Real World Data. He did not have access to historical requests, but he did use historical schedules. From them he fabricated "fictional" requests upon which to test, varying the time windows. He found that wider time windows provide more flexibility and more combinations of schedules. (p.101)

2. He solved test problems from DARP and CVRP literature to test his algorithm's ability to generalize. (p.101)

3. He tested on randomly generated problems. (p.98)

4. He solved a relaxed LP problem to determine an upper bound, and compared his heuristic to it. He said it usually reached the same upper bound. In some cases where he did not find the upper bound, he determined that the bound was not feasibly attainable. (p.102)

5. He compared constrained (hard times) and unconstrained (soft times) problems, and found unconstrained take up to about 8 times more CPU time. (p.107)

6. He found that Early Pickup First (EPF) time ordering generally dominated, but for schedules supporting the same number of legs the flight times were within 5 minutes regardless of the ordering strategy. (p.108-110)

7. He found that clusters of up to 10 requests helped increase the number of requests supported. (p.113)

8. Some of his tests failed to reach the upper bound. (p.116)

9. Some tests failed to converge to a solution within 45 minutes, so he concluded some indication of solution progress and method of aborting the solution would be useful. (p.117)

10. He suggests that schedulers must use realistic time windows, since that will help both the traveler and the algorithm. (p.119)

11. He concluded that the Air Force's concept of a "mission" to package an aircraft's itinerary over a period of time may need reexamination. He determined that "it is important to give the scheduler the ability to negotiate the operational bounds for the selected missions within the constraints of the fleet schedule structure." (p.120)

12. He recommends longer missions be broken into smaller missions based on the solution, since most long missions traverse the home base anyway, in order to satisfy crew duty constraints. (p.121)

13. One comparison used 3 aircraft and 30 hours 38 minutes, versus the expert-produced 5 aircraft and 34 hours. The number of aircraft required was determined by overlapping missions. (p.122)

14. He tested an existing schedule by unscheduling all requests, and rescheduling them as a batch. His solution required 2 fewer aircraft and a 9 hour (13%) reduction in scheduled flight time. (p.123)

15. He found he could use it to effectively reschedule requests following a simulated schedule failure (i.e. aircraft maintenance abort). He used "dummy" requests to force the program to retain the flight legs which had already occurred (before the aircraft abort), and rescheduled the remainder of the requests onto the remaining missions. (p.125)

16. When he applied the algorithm to a DARP, he found that computational requirements grew rapidly. In attempting to minimize the number of soft time window violations (which are not of primary importance, or else they would be hard time windows), we sacrifice solution quality in terms of minimum tour length. (p.125-127)

17. The CVRP has no time windows; since SchedGen relies on exploiting the time windows, he found that the algorithm was computationally inefficient. (p.125-131)

18. Walker recommended using published test sets to test DAKOTA. He used the Eil22 CVRP test set, making necessary modifications to his heuristic in order to solve the set. He reported a solution tour length of 461, but did not give an indication of the solutions obtained by other existing heuristics. (128-131)

Appendix B: Difficulties with DAKOTA

This appendix is divided into two areas. The first section gives a chronology of the general problem areas we encountered in trying to install and use the DAKOTA program. The second section gives specific details of the problems or errors experienced trying to use DAKOTA to perform any scheduling.

Chronology of Installation Problems

Prior to installing DAKOTA for use on AFIT's computers, I contacted Doug Schesvol at North Dakota State University. He compiled the program into a single "tar" file for us to download. Initially when we downloaded (by FTP) the tar file containing the program elements, we found that the database was not included. We contacted Schesvol, who compiled the remainder of the necessary files for our installation. Since DAKOTA was designed to run under Solaris, we installed it on a Sun-10 with 64 MB of memory, running Solaris 2.3.

The program came with an extensive database which had been used at NDSU. It included a number of aircraft, missions, and requests already scheduled. The schedules were for fictional flights in October 1996. Since USAFE has fewer OSA aircraft now than when NDSU developed their database, I attempted to delete several aircraft tail numbers and types from the database, but I was unable to do so. The program *correctly* informed me that I needed to delete aircraft tail numbers before I could delete an aircraft type (e.g., C-135). When I tried to delete the aircraft by tail number, the program *correctly* advised me of the necessity to delete any missions assigned to that aircraft

before I could delete it. When I tried to delete the missions, the program again *correctly* reminded me that I needed to unschedule any requests which had been scheduled onto those missions. These warnings are all "normal" and I anticipated them. Unfortunately, after I unscheduled *all* of the requests, deselected the requests, and removed the missions from the database, I was still unable to delete *some* of the aircraft tail numbers, and *all* of the aircraft types.

I described these problems to Schesvol, and he could not explain the problems. He created another database, this time containing the same aircraft, but without the previously scheduled requests included. This time I was able to successfully delete all the tail numbers and aircraft types which I did not require.

Unfortunately, I noticed another database problem. When an arrival time at a destination is entered, the DAKOTA program is designed to automatically calculate the required departure time from the origin in order to make the required arrival time. The departure time includes an "earliest departure" time and a "latest departure" time, based on the speeds of the slowest and fastest aircraft in the database, respectively. After I deleted the H-1 helicopter from the database, DAKOTA apparently still calculated the earliest departure times based on the airspeed of an H-1, *even though that aircraft no longer existed in the database.*

Next, I attempted to define missions. I was able to define a few missions, but when I tried to define a mission for *some* tail numbers for *some* time periods, I got a warning message stating "Unable to allocate requested aircraft for given time interval." After explaining the problem to Schesvol, he made a correction to the program allowing

the database server to relax the time lock on the aircraft. His fix *seems* to have worked; I was unable to duplicate that particular problem again.

After installing his correction, I again attempted to define a few token missions and requests and to schedule them to determine if the program was working correctly. It successfully accepted the mission and request definitions, and successfully scheduled them. Since it appeared that the program was working appropriately, I entered several more missions and request definitions, but I got a number of other errors. I was unable to successfully schedule any other requests, and I was unable to save the current solution to the database. When I described this problem to Schesvol, he asked me to FTP our database to him for his examination. He reported "I was not able to pull up any requests, missions, passengers, etc. from the database that you ftp'ed. However, I was able to dump the data to ASCII files using a database utility, so it looks like there was data in the database. I was able to reload your data and things seem to work now." When I then FTPed the database *back* from him and installed it, it *seemed* to work perfectly for a short time, but then failed again.

Specific Problem Areas

These first three problems were discovered *before* installing Schesvol's update. I have not seen them recur since the update was installed, but I was not able to get enough experience to determine for certain that the problems had been fixed:

1. Several times I tried to define a mission (ex: 1/Feb/95 0600-2300 for tail number 40087) but I got an error "Unable to allocate requested aircraft for the given time

interval." There were no other missions in the database for that aircraft. I tried an alternate tail number and it worked.

2. With that mission defined, I tried to schedule a request for a "short" round-trip flight (about 4 hours, including layover) to arrive at 1500 (i.e. flight should be about 1100-1500). I got a warning that "Scheduled request leg does not fit on any selected missions." I tried to "force it" manually by adding legs to the mission, but when I tried to append a stop, I got an error "Must enter origination ICAO", but it would not let me.

3. I defined a new mission. Then I tried to retrieve the mission, but I got a dialog box warning "Unable to obtain time lock."

The next items are problems I have had since making the update:

1. I could see a defined mission when I hit the "Next Page" button on the Schedule page, but when I tried to retrieve one, it was not listed in the database. That happened whether or not I had saved the database.

2. I deleted all the requests, then missions, then tail numbers, and then aircraft type for C-12, C-135, and H-1, but for a "short leg" of about 1-1/2 hours for the remaining C-20s, C-21s, or T-43s, the program automatically figured the time window to take up to about four hours.

3. On several occasions I witnessed DAKOTA scheduling missions which seemed to defy attempting to minimize the flight time and distance. One example follows:

    a) I considered two requests: The first was routed from Aviano AB Italy, to Cairo Egypt, to Tel Aviv Israel. The second was for the next day, from Sigonella Italy

(Sicily), to Aviano Italy, and then Ramstein AB Germany. I defined a mission which covered the two-day span of both requests, expecting that the Decision Support System (DSS) in DAKOTA would schedule both requests onto that one mission, scheduling the aircraft to remain overnight in Israel. Instead, it routed that aircraft to continue on to Warsaw Poland, and scheduled another aircraft the next day to deadhead to Sicily, and then on to Aviano and back to Ramstein.

b) There were no other requests or missions which would require that circuitous routing to minimize flight time.

4. I could not save a solution to the database. I got a message stating "Mission must have at least one leg. Solution not saved." This message was after I had already scheduled several requests onto the mission. I *was* able to save solution to a file.

5. On *numerous* occasions, the DSS would declare that a request would not fit onto "any selected mission", even though there were missions which clearly could support the request.

6. On some occasions, especially on missions which span multiple days, it left the first origin blank, instead of starting at Ramstein. The origin should have been the home base of Ramstein. For example:

```
------    Dest 1

Dest 1    Dest 2

Dest 2    ------
```

7. One request was for four foreign national officers to fly from Warsaw, to Ramstein, to Spangdahlem AB Germany, and back to Warsaw. Instead of dedicating an aircraft

and mission to support this, the DSS scheduled two of the three legs on one mission, and used another mission to go from Ramstein, to Sigonella, to Aviano, to Spangdahlem, and to Ramstein. One problem is that the routing would require the contingent to fly the whole route just to travel the half hour distance from Ramstein to Spangdahlem. The second problem is that if the passengers *did* fly the long way to get to Spangdahlem, it would prevent them from *simultaneously* flying on the leg from Warsaw to Ramstein.

8. On one test, I entered 29 requests (covering 8 days) and three missions (covering the first two days) and ran the automated Decision Support System (DSS). It scheduled *all* 29 requests, with several violations, on the 3 missions spanning 1-8 Feb—in violation of the missions' definitions. I noted some Soft Time Window violations (which are acceptable), and several Hard Time Window violations. I did not notice any endurance, capacity, or turntime violations.

   a) Again, the remarkable thing was: A mission defined *only* for 1 Feb was tasked to support requests for the next *eight days*. Additionally, there was another mission defined for the next day (2 Feb) which was *not* scheduled to fly to support any requests. When I tried to manually schedule the 2 Feb request onto the 2 Feb mission, *I could not*. Instead, I got a message that it would not "fit on any selected mission."

9. I could not modify a request which had been scheduled (or unscheduled) but not yet saved to database. I wanted to change the 1200 departure to a hard time, but could not, not in Schedule window with Definition/Select Request menu, and not in

Requests window. In each case, I got a "Request not found in database" message. So I deselected the request and added a brand new request, this time with a hard departure time of 1200. When I tried to manually schedule that request, I got a "Selected request leg does not fit on any selected missions" message. In order to schedule for those requests, I had to: 1) Manually enter the mission legs. 2) Modify those two legs to force it to depart and arrive at the desired request times. Ultimately, it worked, but it resulted in the next request's takeoff time (1225) being a hard violation. If I modified *that* mission, then the *next* leg had a Hard Time Window violation, even though that was not a defined *request*—it was a leg DAKOTA inserted for a fuel stop on the way to another stop in Turkey. The leg in Turkey was not even requested until the next day (2 Feb). Besides, that request was for a courier from Ramstein to Aviano, etc., with the courier on board from Ramstein to Aviano and on subsequent legs. If it scheduled the previous day's mission to continue on that day, then it would prohibit fulfilling the courier's request, which was to fly the mission on a single day.

a) I had unscheduled the request from all missions, but I still could not modify it. I had to deselect (i.e. delete) it entirely, and redefine a new one to make a minor change to the first.

b) The problem was, it considered an automatically-defined fuel stop on the way to Turkey a Hard Time Window violation—even though that request was not supposed to depart for Turkey until the next day! The Courier was not going to

leave Ramstein until the next day. There were *no* requests which drove it to fly to Italy and on to Turkey *that* day.

10. In order to schedule one request, I added a new mission on 2 Feb (0600-2000). When I attempted to manually schedule that request, I got a "Selected request leg does not fit on any selected missions" warning. I defined a new mission tail number, with same result. I deselected that mission, and added another new one, with the same result, still unable to schedule the request. A consequence of having a mission with no legs on it is that I was unable to save the schedule to a file, but not to the database.

   a) I tried to define the legs manually but could not. I tried to use the DSS to define the legs, but it ignored that mission completely, scheduling the requests onto the previous day's mission anyway. At that point, I was out of luck. I could not save to the database until I defined a leg for that mission; I could not define the legs manually; I could not define *any* legs *on that mission* with the DSS, since it used an expired mission (from the day before) instead. Therefore, I *could not save to the database.*

11. Since I found I was unable to add a new mission when the requests were already defined, I tried deleting all missions and requests, starting from scratch, this time adding missions first. The problem with this is that I may not know if an additional mission is required until I have tried to schedule requests and found that they will not fit on existing missions. After I entered a number of requests, I found I was *unable* to define any additional missions to support them.

12. After that, I gave up and reloaded a solution saved to a file. I tried to define a mission for 2 Feb, got an "Unable to allocate requested aircraft for the given time interval" message for two different tail numbers. Both of these aircraft were only defined on a mission from 0600 to 2000 on 1 Feb, so they should be available for the new mission definition *anytime* the following day.

13. Once again, I tried to run the DSS, and once again it scheduled all the requests for eight days onto two 14-hour missions. Once again, I tried to manually schedule, and once again, it said "Selected request leg does not fit on any selected missions." I could not define the legs either by the DSS or manually.

14. I then tried: 1) Defining the mission twice more, with different tail numbers, and rescheduling without success, and 2) Defining another new mission, with yet another tail number, for a three-day period (1-3 Feb) spanning the single day (2 Feb) for which I was trying to schedule the request. Once again, it declared "Selected request leg does not fit on any selected missions." Again, I could not define the legs either by the DSS or manually.

15. The only other option was to define *each leg* on that mission manually. I attempted to "append" stops to that leg, but it would not allow that, giving me a "Must enter origination ICAO" message. It allowed me to enter origination date and time, but not ICAO; I could not make the ICAO field active by clicking in it. Thus, it would not allow me to define individual mission legs.

16. On numerous occasions the database server would fail, requiring me to restart the server again, a 15 minute process. Most times I could not determine what actions

may have caused the server to fail. One time which I *could* determine was when I

accidentally entered an origin ICAO as "EDUL " (with a space), which caused a

"Segmentation Fault" and closed the Requests screen. When I tried to run Requests

again, I got an "Unable to connect to Database Server" error. Any unsaved work

would have been lost.

17. On other occasions, the screen and program simply locked up, requiring the terminal

to be rebooted.

Appendix C: Program General Comments and Suggestions

Anthony Schooler, AFIT/SC (Computer Section), gave me a great deal of assistance in working with and installing DAKOTA. He had several comments about the program:

1. He felt that a different database server might work better. He speculated that the reason for the crashes or locks might be from conflicts between the database server and the display system. He felt that UNIX database libraries like "dbm" might work reliably.

2. He *did* like the index and table file system used by the database. He felt it would be a faster system than trying to load the whole database into memory when needed.

3. Since it uses a single database server, he was concerned that when the server locked up it could have damaged the database structure or index files. He thought a database system which could rebuild the index files from scratch might eliminate some problems.

4. He suggested that different users should have different working areas for saving their solutions, to prevent the potential for overwriting each other's solutions.

5. He noted that many of the directories were "hard coded" in the scripts, requiring the directories to be redesignated in the script layout.

6. Originally AFIT's NFS file system caused conflicts with permissions. We tried different combinations of permissions and groups, even ultimately making my

personal account the owner of all files, but still had the database conflict problems described above.

7. He speculated that some of the operating problems may have come from a mismatch of the libraries in the operating systems. Apparently NDSU uses Solaris 3.5, compared to AFIT's Solaris 2.3, but Dr Nygard wrote in electronic mail on 18 July 1995, "Any SUN running Solaris 2.X should run it seamlessly."

8. He thought that users should be supplied with the tools to access or fix the database. One example is the utility which Schesvol used to export the database to an ASCII file.

9. Finally, when I told him that Schesvol had referred to this version of DAKOTA as a "beta" version, Schooler thought that was an *overstatement* of its status, based on the problems we had encountered.

I would like to offer the following suggestions for improving the program in future revisions:

1. Currently, the program requires a set of requests be assigned to previously defined missions, instead of the other way around. A conceptual change to the approach would be helpful to a scheduler in minimizing the number of missions (i.e., aircraft and crews) required. If the program prompts for a new mission to be defined, or even gives suggestions for a mission start and stop time which would match the greatest number of requests, it would be helpful to the scheduler.

2. The program needs to account for crew duty day restrictions. Currently, it allows the scheduler to define a mission for any period length, without consideration to the

maximum duty day length. This is useful for defining a "Remain Overnight" mission (e.g., leave on Monday, return on Thursday), but it still needs to consider that the crew can only fly for a maximum number of consecutive hours within that period without a break long enough to allow for crew rest.

3. The "hyperhelp" file gives some useful information, but not *that* much more than is already included in the tutorial and manual. It is a *huge* file, over 48 MB, since it is mostly graphical. Even the "text" is saved as a graphic file, so it is very slow. It took me around a minute to jump to each hyperlinked page.

4. The program comes with a main menu system, but it is not yet operable. This would be a useful means of accessing the numerous different screens DAKOTA requires.

5. Like Schooler, I would recommend that the utilities used at NDSU be both available and documented for users at different locations.

6. Many of the times the travelers are not particular about their itineraries. For example, many of the passengers in USAFE are couriers who might need to specify several destinations (in order) that they need to visit on a certain day, but they may not be concerned about specific departure or arrival times. Currently, DAKOTA requires the scheduler to define time windows for each departure and arrival. An "As Required" option for the time windows might increase the computational complexity, but it would allow the scheduler greater flexibility, and would likely result in more efficient schedules in terms of flight time and distance.

7. Similarly, if it is hoped that this program be extended to scheduling for theater airlift, it will need to account for the cargo mission. For cargo flights, particular time windows might not be as important as loading order.

8. DAKOTA needs some quick or easy way to delete data (aircraft, passengers, requests, etc.).

9. It also needs some effective means of producing an output for the scheduler. A print option was recently installed on a menu, but the scheduler ought to have some means of customizing output reports.

10. DAKOTA needs to account for some aircraft having different home bases. That means not simply defining a new default home base, but defining a different home base *for each aircraft*.

11. In order to aid in evaluating how efficient a schedule is, DAKOTA should be modified to provide an output of the amount of flight time required by a schedule.

12. When the scheduler manually changes the departure time for a scheduled leg, he then has to manually change the subsequent arrival time too. It would be convenient if it would automatically update the arrival time based on the estimated enroute time, while still allowing the scheduler to override that time if necessary.

Appendix D: Historical USAFE Data

Table 5
USAFE OSA REQUESTS
February 1995

Arr Date/Time	Location	Dep Date/Time	# PAX
	Ramstein AB, GE (EDAR)	1/0745	3
1/0830	Laarbruch AB, GE (EDUL)	1/1200	
1/1245	Ramstein AB, GE (EDAR)		
	Rota, SP (LERT)	1/1300	6
1/1600	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	1/1300	7
1/1500	Rota, SP (LERT)		
	Aviano AB, IT (LIPA)	3/AR	1
3/AR	Capodichino, Naples, IT (LIRN)		
	Ramstein AB, GE (EDAR)	2/AR	Courier
2/AR	Akrotiri, Cypress (LCRA)	2/AR	
2/AR	RAF Mildenhall, UK (EGUN)		
	Ramstein AB, GE (EDAR)	3/0700	3
3/AR	Aviano AB, IT (LIPA)	3/AR	5 (+2)
3/1700	Sigonella, IT (LICZ)		
	Vilnius, Lith (UMWW)	4/AR	7
4/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	3/AR	Courier
3/AR	RAF Mildenhall, UK (EGUN)	3/AR	
3/AR	Capodichino, Naples, IT (LIRN)	3/AR	
3/AR	Ramstein AB, GE (EDAR)		
	Moron AB, SP (LEMO)	4/AR	7
4/AR	Ramstein AB, GE (EDAR)		
	Aviano AB, IT (LIPA)	5/1615	3
5/AR	Ramstein AB, GE (EDAR)		
	Aviano AB, IT (LIPA)	5/0400	6
5/AR	Cairo, EG (HECA)	5/1030	
5/1130	Ben Gurion, IS (LLBG)		
	Warsaw, PO (EPWA)	6/0900	4
6/1115	Ramstein AB, GE (EDAR)	7/0830	
7/AR	Spangdahlem AB, GE (EDAD)	7/1430	
7/1645	Warsaw, PO (EPWA)		
	Sigonella, IT (LICZ)	6/1320	5 (2)
6/1500	Aviano AB, IT (LIPA)	6/AR	3 (-2)
6/1750	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	6/0700	6
6/AR	Diyarabkir, TU (LTCC)	6/1230	
6/AR	Incirlik, TU (LTAG)	9/0800	
9/AR	Ramstein AB, GE (EDAR)		

Arr Date/Time	Location	Dep Date/Time	# PAX
	Ramstein AB, GE (EDAR)	7/AR	Courier
7/AR	RAF Alconbury, UK (EGWZ)	7/AR	
7/AR	Wiesbaden, AB, GE (EDOU)		
	Akrotiri, Cypress (LCRA)	7/AR	Courier
7/AR	RAF Mildenhall, UK (EGUN)		
	RAF Alconbury, UK (EGWZ)	8/1000	3
8/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	8/1030	6
8/AR	RAF Northolt, UK (EGWU)		
	Ramstein AB, GE (EDAR)	8/AR	2
8/AR	RAF Alconbury, UK (EGWZ)		
	RAF Mildenhall, UK (EGUN)	10/AR	2
10/AR	Ramstein AB, GE (EDAR)		
	Capodichino, Naples, IT (LIRN)	8/0950	3
8/1100	Gety	8/1600	
8/1635	Pisa, IT (LIRP)	9/1450	
9/1645	Capodichino, Naples, IT (LIRN)		
	Ramstein AB, GE (EDAR)	8/0845	6
8/0900	RAF Mildenhall, UK (EGUN)		
same	RAF Alconbury, UK (EGWZ)	10/1130	6
10/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	9/AR	1
9/AR	Ankara, TU (LTAE)	10/AR	
10/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	9/0700	2
9/1120	Ankara, TU (LTAE)	9/1235	
9/1335	Balikesir, TU (LTBF)	9/1620	
9/1710	Ankara, TU (LTAE)	10/1600	
10/1750	Araxos AB, GR (LGRX)	10/1905	
10/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	10/1655	3
10/AR	Bonn, GE	10/2300	
10/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	10/AR	Courier
10/AR	RAF Mildenhall, UK (EGUN)	10/AR	
10/AR	Capodichino, Naples, IT (LIRN)	10/AR	
10/AR	Zagreb, CR (LDZA)	10/AR	
10/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	14/1600	6
14/AR	Stuttgart, GE (EDOC)		

Arr Date/Time	Location	Dep Date/Time	# PAX
	Ramstein AB, GE (EDAR)	9/07	5
9/0800	Aviano AB, IT (LIPA)	9/1200	
9/1300	Capodichino, Naples, IT (LIRN)	9/1500	
9/1830	Incirlik, TU (LTAG)	10/0800	
10/1000	Cairo, EG (HECA)	10/1200	
10/1345	Soudabay	11/1345	
11/1700	Sigonella, IT (LICZ)	12/1030	
10/1430	Torrejon, SP (LETO)	12/1530	
12/1600	Rota, SP (LERT)	13/1200	
13/1530	Lajes AB, AZ (LPLA)		
	Ramstein AB, GE (EDAR)	9/0700	1
9/0830	Aviano AB, IT (LIPA)		
	Vicenza AB, IT (LIPT)	13/0730	3
13/0830	Capodichino, Naples, IT (LIRN)	13/1400	
13/AR	Vicenza AB, IT (LIPT)		
	Ramstein AB, GE (EDAR)	13/0700	6
13/AR	Tirana, Alb (LATI)	15/1600	
15/AR	Ramstein AB, GE (EDAR)		
	Aviano AB, IT (LIPA)	13/1350	2
13/1455	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	13/0800	2
13/0810	RAF Mildenhall, UK (EGUN)	14/1115	
13/1325	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	14/AR	Courier
14/AR	Incirlik, TU (LTAG)	15/AR	
15/AR	Ramstein AB, GE (EDAR)		
	Aviano AB, IT (LIPA)	14/1500	4
14/1915	Incirlik, TU (LTAG)		
	Ramstein AB, GE (EDAR)	15/0830	6
15/AR	Decimomannu, IT (LEID)	16/1400	
16/AR	Ramstein AB, GE (EDAR)		
	Capodichino, Naples, IT (LIRN)	16/1500	2
16/AR	Villa Franca AB, IT (LIPX)		
	Villa Franca AB, IT (LIPX)	16/1720	4
16/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	16/0750	4
16/1030	Moron AB, SP (LEMO)	16/1400	
16/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	17/0800	2
17/AR	Spangdahlem AB, GE (EDAD)	17/1900	
17/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	17/AR	Courier
17/AR	RAF Mildenhall, UK (EGUN)	17/AR	
17/AR	Capodichino, Naples, IT (LIRN)	17/AR	
17/AR	Zagreb, CR (LDZA)	17/AR	
17/AR	Ramstein AB, GE (EDAR)		

Arr Date/Time	Location	Dep Date/Time	# PAX
	RAF Mildenhall, UK (EGUN)	17/1700	7
17/AR	Ramstein AB, GE (EDAR)		
	Ramstein AB, GE (EDAR)	17/0820	7
17/0830	RAF Alconbury, UK (EGWZ)		
	Ramstein AB, GE (EDAR)	20/AR	6
20/AR	Warsaw, PO (EPWA)	20/AR	
20/1600	Ramstein AB, GE (EDAR)	24/1000	
24/1200	Warsaw, PO (EPWA)		
	RAF Northolt, UK (EGWU)	20/1430	3
20/1900	Incirlik, TU (LTAG)	21/1100	
21/1700	Riyadh, SA (OERY)		
	Aviano AB, IT (LIPA)	21/1810	2
21/1915	Ramstein AB, GE (EDAR)		
	Capodichino, Naples, IT (LIRN)	16/1500	4
16/1600	Villa Franca AB, IT (LIPX)	16/1715	
16/AR	Ramstein AB, GE (EDAR)	21/1500	
21/1630	Villa Franca AB, IT (LIPX)	21/1745	
21/AR	Capodichino, Naples, IT (LIRN)		
	Ramstein AB, GE (EDAR)	21/AR	4
21/AR	Vilnius, Lith (UMWW)		
	Vilnius, Lith (UMWW)	21/AR	6
21/AR	Ramstein AB, GE (EDAR)		
	Capodichino, Naples, IT (LIRN)	21/AR	4
21/1200	Stuttgart, GE (EDOC)	21/1800	
21/AR	Capodichino, Naples, IT (LIRN)		
	Ramstein AB, GE (EDAR)	28/AR	Courier
28/AR	Akrotiri, Cypress (LCRA)	28/AR	
28/AR	RAF Mildenhall, UK (EGUN)		
	Stuttgart, GE (EDOC)	27/1255	5
27/1445	Capodichino, Naples, IT (LIRN)		
	Capodichino, Naples, IT (LIRN)	28/0545	2
28/0655	Aviano AB, IT (LIPA)	2/1630	
2/1740	Capodichino, Naples, IT (LIRN)		
	Ramstein AB, GE (EDAR)	27/AR	4
27/AR	Riga, Latvia (UMRR)	3/AR	
3/AR	Ramstein AB, GE (EDAR)		
	Bucharest, RO (LRBS)	27/AR	4
27/1600	Ramstein AB, GE (EDAR)	3/1000	
3/AR	Bucharest, RO (LRBS)		
	Ramstein AB, GE (EDAR)	26/1400	7
26/1530	Aviano AB, IT (LIPA)	27/1800	
27/1930	Ramstein AB, GE (EDAR)		
	Tallinn, Estonia (ULTT)	24/1600	3
24/AR	Stuttgart, GE (EDOC)		

Arr Date/Time	Location	Dep Date/Time	# PAX
	Ramstein AB, GE (EDAR)	23/1605	1
23/1650	Chievres AB, BE (EBCV)	23/1905	
23/2000	Capodichino, Naples, IT (LIRN)	24/1100	
24/1300	Ramstein AB, GE (EDAR)		
	Capodichino, Naples, IT (LIRN)	24/0700	6
24/0750	Brindisi, IT (LIBB)	24/0900	
24/0945	Amendola, IT (LIBA)	24/1030	
24/1045	Ancona, IT (LIPY)	24/1530	
24/1630	Capodichino, Naples, IT (LIRN)		
	Capodichino, Naples, IT (LIRN)	24/1600	4
24/AR	Stuttgart, GE (EDOC)		
	Ramstein AB, GE (EDAR)	23/0700	6
23/0930	Sigonella, IT (LICZ)	23/1300	
23/1530	Ramstein AB, GE (EDAR)		
	Stuttgart, GE (EDOC)	22/0700	3
22/AR	Tallin, Estonia (ULTT)	22/1700	
22/AR	Stuttgart, GE (EDOC)		

Appendix E: Test Plan

Controlled Experiments

In order to conduct a precise experimental comparison of the two scheduling

methods, a controlled experiment is suggested. The test is to be conducted by comparing

the performance of the DAKOTA program to the performance of a human scheduler. In

order to provide the most meaningful appraisal, resources should be dedicated exclusively

to the testing for duration of the evaluation. The computer running DAKOTA should not

have any other tasks running in the background, and preferably a facility separate from

the schedulers' normal duty station should be used during the testing to minimize outside

interruptions.

Two experienced USAFE schedulers will be required for the testing. Their level

of experience must ensure that they provide a realistic representation of the scheduling

system currently in use. Each scheduler must be trained in the use of DAKOTA and be

given ample opportunity to practice and become comfortable with the use of the program.

They should have any of their questions about the program answered before beginning

the testing. Another impartial, experienced scheduler, while not necessarily directly

involved in the testing, should be available to resolve any questions or disputes.

Task one of the schedulers to produce schedules using USAFE's current system;

task the other to schedule using the DAKOTA system. In order to prevent any scheduler

bias, the two schedulers should be randomly assigned to either of the scheduling systems

for *each* schedule to be completed. That is, for one schedule, Scheduler A might use the

manual scheduling system and Scheduler B the DAKOTA system; on the next sample, they *may or may not* be reversed.

The impartial scheduler/observer should establish a reasonable time limit for the experiments. This time limit will differ for each case, depending on the size of the problem. The purpose for the time limit is to provide a reasonable upper bound on the amount of effort which can be dedicated to solving the problem. In an actual scheduling problem, the upper bound is likely to be the execution time for the scheduled missions themselves.

Analysis of Data

After the testing environment has been arranged, the next issue for the experimenter to consider is the test data. The data collected will be examined for its characteristics; it should be representative of USAFE's operational environment. Once the data is reviewed, a means of choosing a representative sample will be determined.

The experimenter should collect historical data on the number of airlift requests submitted to USAFE for each day over the previous year. The requests should be broken down by the number of requested *legs* for the travel, as that gives a more realistic measure of the level of scheduling effort required. For instance, if a traveler requests transportation from Point A, to Point B, and thence to Point C, that should count as *three* legs.

A one year span should be considered to account for potential seasonality in the number of daily airlift requests. A longer period may be considered if the data suggests a longer cycle.

Once the researcher has the number of requested airlift legs for that time period, an analysis of that data may be performed. The intent of this analysis is to determine what factors may affect the daily workload for the scheduler so that a representative sample may be drawn for the testing. First, the data should be evaluated to determine if there are any trends apparent. A computer program such as Forecast Pro may prove helpful. Any linear trends in the data should be noted. (Has there been a drawdown in the forces in Europe over the period? Has there been an increase in the workload in order to support the Bosnian peace efforts?) Do the data present any seasonality? Of what period? It may seem reasonable to expect to see at least a seven-day period present. (Sundays are likely to be a slow day, while the pace may normally increase on Monday morning.) Are there any other periods present? Is there a change in the rate of flight activity associated with the end of the month or quarter? Do summers, Christmas, or other holidays affect the data? In each case, the expert observer must make a determination as to the nature of any significant changes in the number of requests. Any highly "unusual" data points may be discarded if the experimenter considers them highly unlikely to reoccur, as may be the case with a large, one-time conference which required a great deal of OSA support, or a one-time safety stand-down with no flying at all.

Once the experimenter has used his or her best judgment in eliminating any of the outlying data points, the remaining data can be classified by the number of requested legs for that day in order to facilitate sampling representative data for the testing.

<u>Data Sampling</u>

Several different techniques are discussed in this section to give the experimenter an idea of what sampling methods might be considered. The purpose of sampling is to draw representative data from the population so that reasonable inferences may be drawn from them.

<u>Simple random sampling</u>. In this sampling method, each element in the population has an equal probability of being sampled. This method offers the advantages of not requiring any advance knowledge of the population, freedom from classification errors, and computational efficiency. Its disadvantages include not taking advantage of any advance knowledge about the population and relatively large errors for the sample size. (Zikmund, 1994:379; Davis and Cosenza, 1993:228)

<u>Systematic sampling</u>. Under this sampling method, the first sample is chosen at random, and every *n*th observation thereafter is sampled. This raises the danger of *periodicity*, which could be a factor, particularly if the order of the data is not truly random. For instance, if observations are made every seventh day, it is possible that increased variability could be introduced into the sample if observations on the same day each week are not independent. (Zikmund, 1994:379; Davis and Cosenza, 1993:228)

<u>Stratified sampling</u>. This sampling method involves dividing the population into subgroups or *strata*, and then sampling randomly from within each stratum. The stratification variable is chosen so that it is a "characteristic of the population elements known to be related to the dependent variable or other variables of interest." (Zikmund, 1994:373) Stratified sampling is used when the experimenter suspects an important

characteristic distinguishes between the strata and wants to make sure that the difference is accounted for in the study.

Samples are drawn randomly from within each stratum, so it requires accurate information on the proportion in each stratum. There are three benefits to stratified sampling: 1) Variability in sampling is reduced, 2) We are assured that the sample will not fail to include all of the strata in its representation, and 3) We can estimate characteristics of each stratum for comparison. (Davis and Cosenza, 1993:228-229)

Proportionate stratified sampling is conducted when the number of samples drawn from each strata is proportional to the relative size of the stratum. *Dis*proportionate stratified sampling "is not a problem if the primary purpose of the research is to estimate some characteristic separately for each stratum, and if researchers are concerned about assessing the differences among strata." (Zikmund, 1994:373)

Two primary methods of determining the number of strata for the data are outlined below. The first is expert opinion, whereby an individual knowledgeable about the data determines a meaningful number of strata. The second is the "cumulative square root of the frequency" method. (Scheaffer, Mendenhall and Ott, 1990:128-129) This second method is illustrated through the use of a fictitious example, where the frequency of the number of request legs is recorded, and we seek to determine how to stratify based on the number of legs:

Table 6

Cumulative Square Root of the Frequency

# of Legs	Frequency	$\sqrt{\text{Frequency}}$	Cumulative $\sqrt{\text{Frequency}}$
4	2	1.41	1.41
5	4	2	3.41
6	7	2.64	6.06
7	9	3	9.06
8	5	2.24	11.30
9	3	1.73	13.03
	30		

The principle of this method of stratification is to divide the cumulative frequency square root column scale into equal intervals. Assume we wish to define three strata. Since the cumulative square root of the frequency is 13.03, we would look for the square root of the frequencies within each stratum to be 13.03 ÷ 3 = 4.34 wide. Thus, we would like to have the two stratum boundaries as close as possible to 4.34 and 8.68. On our cumulative scale, we would pick 3.41 and 9.06, reflecting that the strata should be:

Table 7

Strata #1

Stratum	# of Legs
1	4 or 5
2	6 or 7
3	8 or 9

For another example, if we want to divide the data into 4 strata, each stratum would be 13.03 ÷ 4 = 3.26 wide. Our stratum boundaries should be as close as possible to 3.26, 6.52 and 9.78. On our cumulative scale, we would pick 3.41, 6.06 and 9.06. Our strata would be:

Table 8

Strata #2

Stratum	# of Legs
1	4 or 5
2	6
3	7
4	8 or 9

Stratified random or proportional sampling provides the three advantages given above, but the primary one for the purpose of testing DAKOTA is that it will assure that the sample will not fail to include all of the strata. This will guarantee that we test over the full range of the number of requests which the scheduler may expect to encounter.

The scheduler must assess the data to determine what variable to use for the stratification. The scheduler's expert opinion will be key, as he or she considers whether to stratify based on the day of the week, each week or month out of the year, or the number of request legs per day or per week, or any other potential strata.

Sample Size

Once the researcher has decided *how* to sample the data for use in the experiment, the number of samples required must be determined. The greater the number of samples, the more confidence we will have in any inferences drawn. Unfortunately, conducting this testing as described entails a considerable commitment of resources, so the experimenter will undoubtedly wish to economize by limiting the sample size as much as possible. The sample size should at least be large enough to invoke the use of the Central Limit Theorem, which tells us that if the observations are independent, identically

distributed random variables with common mean and finite variance, then the sampling distribution of the mean the observations will converge to a standard normal distribution with large enough sample size. Mendenhall, Wackerly and Scheaffer recommend that a sample size of 30 will ensure that the distribution is approximately normal. (Mendenhall, Wackerly and Scheaffer, 1989:319) The means of allocating the 30 samples across the strata will need to be determined. One method, proportional allocation, draws a number of samples for each stratum based on the proportion with which those strata occur in the population. This and other methods of assigning sample sizes to the strata are discussed in texts on sampling, such as Scheaffer, Mendenhall and Ott (1990).

Statistical Testing

We have already determined that the statistical testing should be performed on three main measures of comparison of the two scheduling methods: solution quality (flight time, number of aircraft or crews, number of soft time window violations and infeasible solutions), computational effort (scheduler workload and number of iterations to reach feasibility), and robustness (brittleness and elasticity).

In performing the statistical tests, the experimenter must determine what significance level is desired. The most commonly used "alpha" levels are 0.05, 0.10 or 0.01, but the experimenter is welcome to choose an alpha which reflects the degree of confidence desired. Next, we detail how to perform three different statistical tests, using the flight time as the example.

<u>Flight Time</u>

For this test, calculate the amount of <u>flight time</u> required to satisfy the requests using each of the scheduling methods. Our null hypothesis is that System A and System B produce a schedule which requires the *same* amount of flight time; the alternative hypothesis is that System A produces a schedule requiring greater flight time than System B. (Note that we could have chosen a two-tailed test by defining the alternative hypothesis as "System A and System B produce a schedule which does *not* require the same amount of flight time".)

We conduct hypothesis testing to determine if our null hypothesis is reasonable. We compare the results using three of the non-parametric tests described in Chapter II, starting with the <u>Wilcoxon Signed Rank Test</u>. A simplified example follows, where System A and System B both produce six different schedules. For example, on the first schedule, System A required 12.2 flight hours to complete the schedule, while System B required 12 hours:

Table 9

Wilcoxon Signed Rank Test Example
(Hours of Flight Time)

System A	System B	|Difference|	Rank	Signed Rank
12.2	12	0.2	2.5	2.5
20.4	20	0.4	5	5
18.1	18	0.1	1	1
8.8	9	0.2	2.5	-2.5
14.7	15	0.3	4	-4
10	10	0		
				2

Test Statistic: 2
Critical Value: 1
Conclusion: Reject Null Hypothesis

Our test statistic is the sum of the signed ranks, or 2. We then compare that against the critical value of 1 which can be found in texts on nonparametric statistics (Mendenhall, Wackerly and Scheaffer, 1989:780). For a detailed description of this statistical test, refer to (Golden and Stewart, 1985:210-212).

Our null hypothesis was that the two systems produce a schedule which requires the *same* amount of flight time, but our test suggests that we reject the null hypothesis at an alpha level of 0.05 chosen for this illustration. From this we surmise that the mean flight time from System A is larger than that from System B. Since the object is to *minimize* the flight time, System B would be preferred.

Recall that the Wilcoxon Signed Rank Test requires several assumptions: (Golden and Stewart, 1985:210-212)

a. The data consist of matched pairs (x_i, y_i), with the difference defined as

$$d_i = x_i - y_i$$

b. Each d_i is a continuous random variable

c. The distribution of each d_i must be symmetric

d. The pairs (x_i, y_i) represent a random sample from a bivariate distribution

The researcher needs to consider assumption c. in particular, i.e., that the differences are symmetrically distributed. Plotting the differences in a box plot can give a visual measure of the symmetry of the distribution; calculating the skewness can give a numerical measure.

If the researcher determines that the distribution is too asymmetric, or if the two methods tie too often for a useful comparison with the Wilcoxon Signed Rank Test, then the <u>McNemar Test for Significance of Changes</u> is available for the statistical test. In the McNemar Test, the (X_i, Y_i) pairs are classified into $(1,0)$ or $(0,1)$ pairs (based on which of the two heuristics produces the best result on the measure of interest) according to the table below:

Table 10

McNemar Test for Significance of Changes Classifications

Classification of Y_i

		$Y_i = 0$	$Y_i = 1$
Classification of X_i	$X_i = 0$	Heuristics X and Y Tie	Heuristic Y Outperforms Heuristic X
	$X_i = 1$	Heuristic X Outperforms Heuristic Y	Heuristics X and Y Tie

The cases where the two heuristics tie are discarded. The number of (0,1) and (1,0) pairs are counted for the statistical test. Using our same example from before:

Table 11

McNemar Test for Significance of Changes Example
(Hours of Flight Time)

System A	System B	Pairs
12.2	12	(0,1)
20.4	20	(0,1)
18.1	18	(0,1)
8.8	9	(1,0)
14.7	15	(1,0)
10	10	-

Test Statistic: 3
Critical Value: 1
Conclusion: Reject Null Hypothesis

We eliminated the case where the systems tied, and compare the three (0,1) pairs to the two (1,0) pairs. Since our test statistic is greater than the critical value, at an alpha level of 0.05, we reject the null hypothesis and again conclude that System B outperforms System A in producing a schedule which required the least flight time.

The test statistic and rejection rule both depend on the sample size, and their calculation is fairly complicated. The interested reader may refer to a nonparametric statistics text for details. (Conover, 1980:130-133)

The third statistical test to consider in performing the statistical testing is the <u>Sign Test</u>. The sign test is not as powerful a statistical test as the Wilcoxon Signed Rank Test; however, it does not require the same stringent assumptions as the Wilcoxon Signed Rank Test. (Golden and Stewart, 1985:212) If we are testing two heuristics (X and Y) and assume that they are equally likely to produce a better solution on the measure of interest, i.e., $P[y_i - x_i > 0] = 0.5$, then the probability that n or more positive (or negative, for that matter) differences would be found in m observations is:

$$\sum_{k=n}^{m} \binom{m}{k} (0.5)^k \cdot (0.5)^{m-k} \tag{9}$$

A low probability would lend credence to the supposition that the two heuristics are *not* equally likely to produce the better solution.

Using the same example from above, we again eliminate the tied case, considering only the three positive differences and the two negative differences, we find the probability of three or more positive differences out of the five pairs:

$$\sum_{k=3}^{5} \binom{5}{k} (0.5)^k \cdot (0.5)^{5-k} = 0.5 \tag{10}$$

Obviously, with such a limited sample size this test did not allow us to make any conclusive inferences about the population.

Other Measures of Comparison

We used the three statistical tests described above to thoroughly review how to test the performances of the scheduling methods with regards to the flight time required to execute the schedule produced. Next, we consider the other measures of merit proposed for statistical testing.

Number Of Aircraft Or Crews

The number of aircraft or crews required to effect the produced schedules can be tested simultaneously with the testing of the flight time described above by keeping track of the number of *missions* required to support the requests. The difference d_i between the number of missions required by each of the scheduling methods would be tested for significance using the same statistical tests as described above. Rejection of the hypothesis that both methods require the same number of aircraft and crews would suggest that one of the methods performs better on that measure of merit.

Number Of Soft Time Window Violations

A determination of which scheduling method produces the fewest soft time window violations can be made by counting the number of legs which do *not* meet the (soft) time windows requested by the traveler, and comparing them using the statistical tests described above.

Infeasible Solutions

The number of hard time window violations is a measure of the number of infeasible solutions. This can be statistically tested as described above.

Scheduler Workload

The next series of tests is designed to compare the scheduler workload in constructing the schedules. Since the CPU time is a less meaningful measure in USAFE's scheduling than the schedulers' effort, this test is conducted by timing each of the schedulers' "user time" as they complete their tasks under the test plan's controlled conditions. It is particularly important for the investigator to insure that the experiment is *randomized* in order to avoid any systematic bias between the schedulers.

User time is the time measured from the point when a scheduler commences work *on that particular schedule* until the "flyable" schedule is completed or the task is abandoned as infeasible. The commencement time must account for setup time, including the time to start the computer program. In order to provide the most meaningful appraisal, resources and personnel should be dedicated exclusively to the testing for duration of the evaluation. The computer running DAKOTA should not have any other tasks running in the background, and preferably a separate facility should be used during the testing to minimize outside interruptions.

The time a scheduler spends conferring and coordinating on the telephone, making refinements and improvements to the schedule is also included in the user time, as this is a reasonable subtask of scheduling which affects any scheduling system. Time spent on breaks should not be included; in this context, time when the computer program is computing a schedule should not be considered a "break". Note that time spent collecting the initial information (i.e., requests) need not be considered, as this will be identical for each scheduling system. Any questions about what specific tasks need to be

considered in the user time are resolved by the impartial scheduler/observer. The driving concern is *consistency*, in order to produce a meaningful comparison.

The schedulers can be timed as they solved the previous series of problems. The time required to produce both schedules, manually and through the use of DAKOTA, can be statistically compared using the same statistical tests previously outlined.

Number Of Iterations To Reach Feasibility

This test is more complicated to conduct. As a schedule is being produced, the scheduler will have to try to resolve the soft time window violations. This is done by contacting the passengers or point of contact for the affected requests to determine if an alternative time window is acceptable to them, and then making the appropriate changes to the schedule. Clearly, this will be a time-consuming process. This test counts the *number* of times this process is iterated until the schedule is complete. The same statistical tests used earlier may be applied, but the experimenter should carefully consider if the definitions of iterations are comparable. If a valid definition that applies to both systems cannot be determined, this test should not be conducted.

Brittleness

In order to conduct this test, we take the *completed* schedules already produced in the previous experiments and randomly pick 10% of the request legs and *delay* their requested arrival times by one hour. We then randomly pick another 10% of the request legs and *advance* their requested arrival times by one hour. Without attempting to make any revisions to the schedules to accommodate the changes in the requests, we simply

count the number of *new* soft and hard (i.e., infeasible) time window violations, excluding those violations we already counted when testing the number of soft time window violations and infeasible solutions. The *least* brittle scheduling method is the one which has the smallest *increase* in those measures in response to the perturbations in the requests. The statistical tests are conducted as described above.

Elasticity

Finally, for this comparison we keep the requested departure and arrival *times* the same, but we perturb the time *windows* for the request. For a randomly chosen 10% of the legs we double the size of the time window; for another random 10% we halve its size; for another 10% we convert soft time windows into hard; and for another 10% we convert the hard time windows into soft. Again, without making any revisions to the schedules to accommodate the changes in the request time windows, we count the number of *new* soft and hard time window violations for both scheduling methods and compare them using the statistical tests described earlier.

Summary

As the tests are planned and conducted, USAFE personnel may recognize other elements that they wish to test in the systems. It may be most effective to incorporate these tests with those outlined here. A pilot test (a smaller scale version of the full test) may prove beneficial in "ironing out" any "kinks" in the test plan or its implementation.

While it is recognized that finding the time and personnel to assign to the testing will be difficult, the more controlled an experiment that can be conducted, the higher the confidence one will have in the test results.

Bibliography

Aarts, E.H.L., P.J.M. Van Laarhoven, J.K. Lenstra, and N.L.J. Ulder. "A Computational Study of Local Search Algorithms for Job Shop Scheduling," ORSA Journal on Computing, 6: 118-125 (Spring 1994).

Abara, Jeph, "Applying Integer Linear Programming to the Fleet Assignment Problem," Interfaces, 19: 20-28 (July - August 1989).

Andriole, Stephen J., Editor. Software Validation, Verification, Testing and Documentation. Princeton NJ: Petrocelli Books, 1986.

Bacsi, Zsuzsanna, and Ferenc Zemankovics. "Validation: An Objective or a Tool? Results on a Winter Wheat Simulation Model Application." Ecological Modelling, 81: 251-263 (1995).

Balci, Osman, Editor. Methodology and Validation: Proceedings of the Conference on Methodology and Validation 1987. San Diego: Simulation Councils, Inc., 1987.

Barr, Richard S., Bruce L. Golden, James P. Kelly, Mauricio G.C. Resende, William R. Stewart. "Designing and Reporting on Computational Experiments with Heuristic Methods," Journal of Heuristics, 1: 9-32 (Fall 1995).

Bausch, Dan O., Gerald G. Brown, and David Ronen. "Dispatching Shipments at Minimal Cost With Multiple Mode Alternatives." Journal of Business Logistics, 15: 287-303 (1994).

Bentley, Jon L. "Fast Algorithms for Geometric Traveling Salesman Problems," ORSA Journal on Computing, 4: 387-411 (Fall 1992).

Beyerle, John A., Warren G. Hamblet, and David A. Kane. "Independent Verification and Validation of the Advanced Planning System," Final Technical Report RL-TR-95-42. Rome Laboratory, Air Force Materiel Command, March 1995.

Bodin, Lawrence, and Bruce Golden. "Classification in Vehicle Routing and Scheduling," Networks, 11: 97-108, (1981).

Bodin, Lawrence, and L. Levy. "Visualization in Vehicle Routing and Scheduling Problems," ORSA Journal on Computing, 6: 261-269 (Summer 1994).

Braitsch, R. "A Computer Comparison of Four Quadratic Programming Algorithms." Management Science, 18:11 631-643 (July 1972).

Bramel, Julien, and David Simchi-Levi. "A Location Based Heuristic for General Routing Problems," Operations Research, 43: 649-660 (July-August 1995).

Brownlee, K.A. Statistical Theory and Methodology in Science and Engineering, 2nd Edition. New York: John Wiley & Sons, 1965.

Bruns, Thomas J. LOGAIR and QUICKTRANS: A Model in Combination. MS Thesis, AFIT/GLM/LSM/90S-7. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1990 (AAE-3180).

Burgess, Capt Rob. Headquarters, Pacific Air Forces, Hickam AFB HI. Telephone interview. 1 February 1996.

Camp Systems, Inc. Andromeda FS for Windows 95. Ronkonkoma NY: 1995.

Conover, W.J. Practical Nonparametric Statistics, 2nd Ed. New York: John Wiley & Sons, 1980.

Crowder, Harlan, Ron S. Dembo and John M. Mulvey. "On Reporting Computational Experiments with Mathematical Software." ACM Transactions on Mathematical Software, 5: 193-203 (June 1979).

Davis, Duane, and Robert M. Cosenza. Business Research for Decision Making. Belmont CA: Wadsworth Publishing Company, 1993.

Davis, Paul K. Generalizing Concepts and Methods of Verification, Validation, and Accreditation (VV&A) for Military Simulations. Santa Monica: RAND, 1992.

Department of the Air Force. Operational Support Mission (OSA) Management. Air Force Instruction 13-204. Washington: HQ USAF, 23 May 1994.

Department of the Air Force. Modeling and Simulation (M&S) Management. Air Force Policy Directive 16-10. Washington: HQ USAF, 30 January 1995.

Department of the Air Force. Software Management. Air Force Instruction 33-114. Washington: HQ USAF, 30 June 1994.

Department of the Air Force. Developmental Test and Evaluation. Air Force Instruction 99-101. Washington: HQ USAF, 22 July 1994.

Department of Defense. Operational Support Airlift (OSA). DOD Directive 4500.43. Washington: GPO, 30 October 1985.

Dery, Richard, Maurice Landry, and Claude Banville. "Revisiting the Issue of Model Validation in OR: An Epistemological View." European Journal of Operational Research, 66: 168-183 (1993).

Deutsch, Michael S. Software Verification and Validation: Realistic Project Approaches. Englewood Cliffs NJ: Prentice-Hall, Inc. 1982.

Elmer, Michael R. Issues and Challenges in Validating Military Simulation Models. MS Thesis, AFIT/GSO/ENS/95D-02. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1990.

Finn, Erwin. Sales Manager, Camp Systems Inc., Ronkonkoma NY. Telephone interview. January 1996.

Fischetti, Matteo, Paolo Toth, and Daniele Vigo. "A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs," Operations Research, 42: 846-859 (September-October 1994).

Fisher, Marshall L. "Worst-Case Analysis of Heuristic Algorithms." Management Science, 26: 1-17 (January 1980).

Fitzsimmons, John Jr., and John Walker. A Heuristic Approach to Determining Cargo Flow and Scheduling for Air Mobility Command's Channel Cargo System. MS Thesis, AFIT/GOR/ENS/94M-05. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1994.

"Functional Description of the AMC C2IPS Release 2.0C Software, Final Draft" Intermetrics, Inc. 26 Jan 96.

Gass, Saul I. "Model Accreditation: A Rationale and Process for Determining a Numerical Rating." European Journal of Operational Research, 66: 250-258 (1993).

Gendreau, Michel, Gilbert Laporte, and Rene Seguin. "An Exact Algorithm for the Vehicle Routing Problem with Stochastic Demands and Customers," Transportation Science, 29: 143-155 (May 1995).

Glover, Fred, D. Karney, D. Klingman, and A. Napier. "A Computational Study of Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems." Management Science, 20:5 793-813 (Jan 1974).

Goldberg, David E., Genetic Algorithms: Search, Optimization, and Machine Learning, Reading MA: Addison-Wesley, 1988.

Goldberg, David E., Department of General Engineering, University of Illinois at Urbana-Champaign. "Making Genetic Algorithms Fly: A Lesson From the Wright

Brothers." Address to the Air Force Institute of Technology faculty and students. Air Force Institute of Technology, Wright-Patterson AFB OH 23 August, 1995.

Golden, B. "Shortest Path Algorithms: A Comparison." Operations Research 24:6 1164-1168 (Nov-Dec 1976).

Golden, B. and W. Stewart. "Empirical Analysis of Heuristics", in: Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, (eds.) The Traveling Salesman Problem. Chichester: John Wiley & Sons, 1985, 207-250.

Griggs, Brian J. An Air Mission Planning Algorithm for a Theater Level Combat Model. MS Thesis, AFIT/GST/ENS/94M-5. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1994 (AAK9078).

Hartmann, Christa, Johanna Smeyers-Verbeke, Wim Penninckx, Yvan Vander Heyden, Pieter Vankeerberghen, and Desire L. Massart. "Reappraisal of Hypothesis Testing for Method Validation: Detection of Systematic Error by Comparing the Means of Two Methods or of Two Laboratories." Analytical Chemistry, 67: 4491-4499 (December 15, 1995).

Hecht, Herbert, Myron Hecht, George Dinsmore, Sara Hecht, and Dong Tang. "Verification and Validation Guidelines for High Integrity Systems," Final Technical Report RL-TR-94-201 Vol. 1. Rome Laboratory, Air Force Materiel Command, November 1994.

Hillier, Frederick S., and Gerald J. Lieberman. Introduction to Operations Research, 5th Edition. New York: McGraw-Hill Publishing Company, 1990.

Hooker, John N. "Testing Heuristics: We Have it All Wrong," Journal of Heuristics, 1: 33-42. (Fall 1995).

Ignizio, James P. "On the Establishment of Standards for Comparing Algorithm Performance". TIMS Interfaces, 2: 8-11 (1971).

Ioachim, Irina, Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and Daniel Villeneuve. "A Request Clustering Algorithm for Door-to-Door Handicapped Transportation," Transportation Science, 29: 63-78 (February 1995).

Jackson, Richard H.F., Paul T. Boggs, Stephen G. Nash and Susan Powell. "Guidelines for Reporting Results of Computational Experiments; Report of the ad hoc Committee." Mathematical Programming, 49: 413-425 (1991).

Jackson, Richard H.F. and John M. Mulvey. "A Critical Review of Comparisons of Mathematical Programming Algorithms and Software (1953-1977)." Journal of Research of the National Bureau of Standards, 83: 563-579 (1978).

Jaw, Jang-Jei, Amedeo R. Odoni, Harilaos N. Psaraftis, and Nigel H.M. Wilson. "A Heuristic Algorithm for the Multi-Vehicle Advance Request Dial-A-Ride Problem With Time Windows," <u>Transportation Research, 20B</u>: 243-257 (1986).

Johnson, D.S., and C.H. Papadimitriou. "Performance Guarantees for Heuristics", in: Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, (eds.) <u>The Traveling Salesman Problem</u>. Chichester: John Wiley & Sons, 1985, 145-180.

Johnson, Norman L., and Fred C. Leone. <u>Statistics and Experimental Design in Engineering an the Physical Sciences, Vol. 1, 2nd Edition</u>. New York: John Wiley & Sons, 1977.

Kleijnen, Jack P.C. <u>Statistical Techniques in Simulation, Part I</u>. New York: Marcel Dekker, Inc., 1974.

Kolen, A.W.J., A.H.G. Rinnooy Kan, and H.W.J.M. Trienekens. "Vehicle Routing with Time Windows," <u>Operations Research, 35</u>: 266-273 (March-April 1987).

Koskosidis, Yiannis A., Warren B. Powell, and Marius M. Solomon. "An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Window Constraints," <u>Transportation Science, 26</u>: 69-85 (May 1992).

Landry, Maurice, and Muhittin Oral. "In Search of a Valid View of Model Validation for Operations Research." <u>European Journal of Operational Research, 66</u>: 161-167 (1993).

Laporte, Gilbert, Francois Louveaux, and Helene Mercure. "The Vehicle Routing Problem with Stochastic Travel Times," <u>Transportation Science, 26</u>: 161-170 (August 1992).

Law, Averill M., and W. David Kelton. <u>Simulation Modeling and Analysis</u>. New York: McGraw-Hill, Inc., 1991.

Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. <u>The Traveling Salesman Problem</u>. Chichester: John Wiley & Sons, 1985.

Lemmer, John. Rome Laboratories, Rome NY. Telephone interview. 29 January 1996.

Li, Chung-Lun, and David Simchi-Levi. "Worst-Case Analysis of Heuristics for Multidepot Capacitated Vehicle Routing Problems," <u>ORSA Journal on Computing, 2</u>: 64-73 (Winter 1990).

Lopez-Velasquez, TSgt Phyllis. HQ USAFE/AOS/AOOMS (Air Operations Squadron/Special Airlift). Telephone interview. 22 January 1996.

Maj Maher. HQ USAFE/AOS/AOOMS (Air Operations Sq/Special Airlift), Ramstein AB Germany. Telephone interview. 3 January 1996.

Malandraki, Chryssi, and Mark S. Daskin. "Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms," Transportation Science, 26: 185-200 (August 1992).

McCanne, Randy. The Airlift Capabilities Estimation Prototype: A Case Study in Model Validation. MS Thesis, AFIT/GOR/ENS/93M-13. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1993 (AAJ-7538).

Mendenhall, William, Dennis D. Wackerly, and Richard L. Scheaffer. Mathematical Statistics with Applications. New York: Duxbury Publishing, 1989.

Merrill, David L. Facility Location and Routing to Minimize the Enroute Distance of Flight Inspection Missions. MS Thesis, AFIT/GST/ENS/89M-13. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 1989 (AAJ-6204).

Miser, Hugh J. "A Foundational Concept of Science Appropriate for Validation in Operational Research." European Journal of Operational Research, 66: 204-215 (1993).

Murtha, Poul J. "Operations Research and the Airline Industry: A Survey of Current Literature." Report to Dr. Richard Deckro, Professor of EMGT 506, Portland State University, 22 February, 1995.

MSgt Noble. AMC TACC/XOFS, Scott AFB IL. Telephone interview. 1 February 1996.

Nygard, Kendall E., Paul Juell, and Nagesh Kadaba. "Neural Networks for Selecting Vehicle Routing Heuristics," ORSA Journal on Computing, 2: 353-364 (Fall 1990).

Oreskes, Naomi, Kristin Shrader-Frechette, and Kenneth Belitz. "Verification, Validation, and Confirmation of Numerical Models in the Earth Sciences." Science, 263: 641-646 (4 February 1994).

Ortúzar, Juan de D., and Luis G. Willumsen. Modelling Transport, 2nd Edition. Chichester: John Wiley & Sons, 1994.

Pellissier, Renee. System Department, Navy Air Liaison Office (NALO), NAS New Orleans LA. Telephone interview. 21 February 1996.

Perry, William E. How to Test Software Packages: A Step-by-Step Guide to Assuring They Do What You Want. New York: John Wiley & Sons, 1986.

Pooley, John. "A Vehicle Routing Algorithm for the Less-Than-Truckload vs. Multiple-Stop Truckload Problem." Journal of Business Logistics, 13: 239-258 (1992).

Potvin, Jean-Yves. "The Traveling Salesman Problem: A Neural Network Perspective," ORSA Journal on Computing, 5: 328-347 (Fall 1993).

Psaraftis, Harilaos N. "An Exact Algorithm for the Single-Vehicle Many-to-Many, Dial-a-Ride Problem with Time Windows," Transportation Science, 17: 351-357 (Aug 1983).

Rardin, Ronald L., and Benjamin W. Lin. "Test Problems for Computational Experiments—Issues and Techniques". In Mulvey, J. (ed.), Evaluating Mathematical Programming Techniques. Berlin: Springer-Verlag, 1982, 8-15.

Reeves, Colin R. Modern Heuristic Techniques for Combinatorial Problems. New York: John Wiley & Sons, Inc., 1993.

Richter, Hulmut. "Thirty Years of Airline Operations Research," Interfaces, 19: 3-9 (July - August 1989).

Russell, Robert A. "Hybrid Heuristics for the Vehicle Routing Problem with Time Windows," Transportation Science, 29: 156-166 (May 1995).

Rykiel, Edward J., Jr. Letter to Science, 264 330-331 (15 April 1994).

SSgt Brooke, OSCOM (Operational Support Command), U.S. Army, Ft Lee VA. Telephone interview. 22 February 1996.

Savelsbergh, Martin W.P. "The Vehicle Routing Problem with Time Windows: Minimizing Route Duration," ORSA Journal on Computing, 4: 146-154 (Spring 1992).

Savelsbergh, Martin W.P., and M. Sol. "The General Pickup and Delivery Problem," Transportation Science, 29: 17-29 (February 1995).

Schank, John F., James P. Stucker, Michael G. Mattock, and Jeff Rotheberg. New Capabilities for Strategic Mobility Analysis, Executive Summary. Santa Monica: RAND, 1994.

Scheaffer, Richard L., William Mendenhall, and Lyman Ott. Elementary Survey Sampling. Boston: PWS-Kent Publishing Company, 1990.

Sklar, Michael G., R.D. Armstrong and S. Samn. "Heuristics for Scheduling Aircraft and Crew During Airlift Operations." Transportation Science, 24:1 63-76 (Feb 1990).

Smith, Douglas R., Eduardo A. Parra, and Stephen J. Westfold. "Synthesis of High-Performance Transportation Schedulers," Palo Alto: Kestrel Institute, 9 March 1995.

Smith, Douglas R., and Eduardo A. Parra. "Transformational Approach to Transportation Scheduling," Palo Alto: Kestrel Institute, 1993.

Solanki, Rajendra S., and Frank Southworth. "An Execution Planning Algorithm for Military Aircraft," Interfaces, 21: 121-131 (July-August 1991).

Solomon, Marius M. "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints," Operations Research, 35: 254-265 (March-April 1987).

Solomon, Marius M., and Jacques Desrosiers. "Survey Paper: Time Window Constrained Routing and Scheduling Problems," Transportation Science, 22: 1-13 (February 1988).

Steedley, Melvin, Lt Cdr. System Department, Navy Air Liaison Office (NALO), NAS New Orleans LA. Telephone interview. 8 February 1996.

Subramanian, Radhika, Richard P. Scheff, John D. Quillinan, D. Steve Wiper, and Roy E. Marsten. "Coldstart: Fleet Assignment at Delta Air Lines," Interfaces, 24: 104-120 (January-February 1994).

Swenson, Elizabeth J. The Vehicle Routing Problem With Time Constraints. MS Thesis, North Dakota State University, Fargo ND, 1986.

Teodorovic, Dusan., and Goran Stojkovic. "Model to Reduce Airline Scheduling Disturbances," Journal of Transportation Engineering, 121: 324-331 (July/August 1995).

Vander Wiel, Russ J. and Nikolaos V. Sahinidis. "Heuristic Bounds and Test Problem Generation for the Time-Dependent Traveling Salesman Problem," Transportation Science, 29: 167-183 (May 1995).

Walker, Richard S. Heuristics for Customized Passenger Airlift Scheduling. PhD dissertation. North Dakota State University, Fargo ND, October 1994.

Wilkinson, Nita. HQ USAFE/DON, Ramstein AB Germany. Personal interview. 18 October1995.

Wilkinson, Nita. HQ USAFE/DON, Ramstein AB Germany. Electronic mail. 8 February 1996.

Zanakis, S. "Heuristic 0-1 Linear Programming: An Experimental Comparison of Three Methods." <u>Management Science, 24:1</u> 91-104 (September, 1977).

Zanakis, Stelios H., James R. Evans, and Alkis A. Vazacopoulos. "Heuristic Methods and Applications: A Categorized Survey." <u>European Journal of Operational Research, 43</u>: 88-110 (1989).

Zikmund, William G. <u>Business Research Methods</u>. Fort Worth TX: The Dryden Press, Harcourt Brace College Publishers, 1994.

<u>Vita</u>

Capt Keith H. McCready was born on 29 July 1962 in Tawas City, Michigan. He graduated from Tawas Area High School in 1980, and enrolled in the Air Force Academy in Colorado Springs, Colorado. Upon graduating in May 1984, he entered Undergraduate Pilot Training (Helicopter) at Ft. Rucker, Alabama, where he was the distinguished graduate in May 1985. His subsequent assignments have included Eglin AFB, Kirtland AFB, and Misawa AB, Japan, and included flying combat rescue, special operations, and teaching at the Air Force's formal schoolhouse for helicopters. He earned the Master of Science degree in Industrial Engineering from New Mexico State University. In August 1994, he entered the Graduate School of Engineering, Air Force Institute of Technology.

Capt McCready is married to the former Susan Horvath of Johnstown, Pennsylvania. They are the proud "parents" of Kaiser the puppy.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March, 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
PROCEDURES FOR TESTING DETERMINISTIC
SCHEDULING MODELS: A DAKOTA CASE STUDY

5. FUNDING NUMBERS

6. AUTHOR(S)

Keith H. McCready, Capt, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
2750 P Street
WPAFB, OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/96M-08

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ USAFE/DON
Unit 3050 Box 15
APO AE 09094-5015

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The DAKOTA scheduling system has been proposed for use in the United States Air Forces Europe's (USAFE's) Operational Support Airlift (OSA) scheduling. This thesis examines the OSA scheduling topic and reviews the relevant literature on vehicle routing, concluding that exact methods are intractable for large problem sizes. Consequently, heuristic methods must be considered. This thesis takes a detailed look at the DAKOTA heuristic. It examines the concepts of Validation, Verification and Accreditation (VV&A), particularly as they apply to heuristics and algorithms. It then defines what measures of performance may prove useful in judging heuristics and algorithms in general, and details the statistical tests which can be used to make those comparisons. It discusses four of the predominant airlift scheduling models currently in use, and finally develops a methodology which can be used to evaluate a deterministic passenger airlift scheduling heuristic, using DAKOTA as a case study.

14. SUBJECT TERMS

Passenger Routing, Heuristic Methods, Algorithms, VV&A, Testing

15. NUMBER OF PAGES
153

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18